A Database Publication

Computing with the AMSTRAD

No. 6
June 1985
£1

EasyDraw
Complete paintbox program in an 8 page supplement

The independent magazine for CPC464/664 users

Rescue Esmeralda from the tower

Create colourful disc menus

Inspect your micro's memory

Amstrad teach-in

Learn about loops and conditional operators . . .
Investigate the Z80 register pairs . . .
collision detection and screen boundaries . . . and join the sound queue

WELCOME TO THE WORLD OF DATACOM

EXECUTION - They told me it would be bad but I never thought it would be like this...must keep a clear mind... can't afford to panic...time is fast running out...don't think my nerves will stand much more of this!!! A brain straining memory bashing game of words.

SNAIL PACE - Unbearable excitement. 30 'thoroughbred racing snails' battle to the finish. Super features include computer calculated odds, form guide, excellent sound effects, smooth m/c graphics. Rivetting excitement guaranteed.

INTERPLANETARY TRADER - The most addictive, mind blowing, feature packed space adventure you will ever experience on your AMSTRAD CPC 464. Defend your cargo against space pirates...navigate asteroids, black holes, magnetic storms...Total concentration is required in your quest to become a GALACTIC MEGABILLIONAIRE!!!

DATACOM PUBLICATIONS:
407F Hockley Centre,
Birmingham B18 6NF,
Tel: 021-233 1800

Trade enquiries welcome

All games just

£5.95

Each

These games will be available from your dealer or **POST FREE*** direct from: DATACOM PUBLICATIONS using the coupon provided (or just write in)

* U.K. only - Overseas add £1 p & p

Name.....

Address:.....

.....

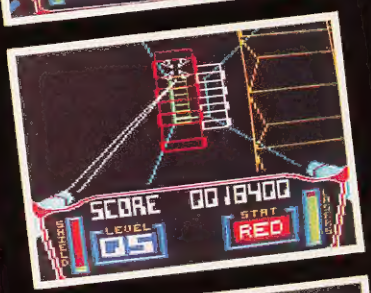
Tel.....

Please rush me ☐ Tapes of Interplanetary Trader
☐ Tapes of Execution
☐ Tapes of Snail Pace

I enclose a cheque/P.O. for £.....

REALTIME SOFTWARE

3D STARSTRIKE



Actual Screen Shots (Amstrad)

AMSTRAD CPC 464 £6.95

ZX SPECTRUM 48K £5.95

- ★ STEREO SOUND
- ★ FAST 3D GRAPHICS
- ★ GREAT VALUE FOR MONEY

PLEASE RUSH ME!

☐ Starstrike (Amstrad) £6.95

☐ Starstrike (Spectrum) £5.95

Name .....

Address .....

.....

Cheques/P.O.s to:

Realtime Softwear, Prospect House, 32 Sovereign Street, Leeds LS1 4BT

AMSTRAD STARSTRIKE

In the shops NOW!

Vol. 1 No. 6 June 1985

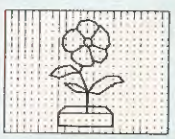
Managing Editor: **Derek Meakin**
Features Editor: **Peter Bibby**
The A Team: **Mike Bibby**
Alan McLachlan
Kevin Edwards
Roland Waddilove
Production Editor: **Peter Glover**
Layout Design: **Heather Sheldrick**
News editor: **Mike Cowley**
Advertisement Manager: **John Riding**
Advertising Sales: **Margaret Clarke**
Editor in Chief: **Peter Brameld**

Editorial: 061-456 8835
Administration: 061-456 8383
Advertising: 061-456 8500
Subscriptions: 061-480 0171
Telex: 667664 SHARET G
Prestel Mailbox: 614568383

Published by:
Database Publications Ltd,
Europa House, 68 Chester Road,
Hazel Grove, Stockport SK7 5NY.

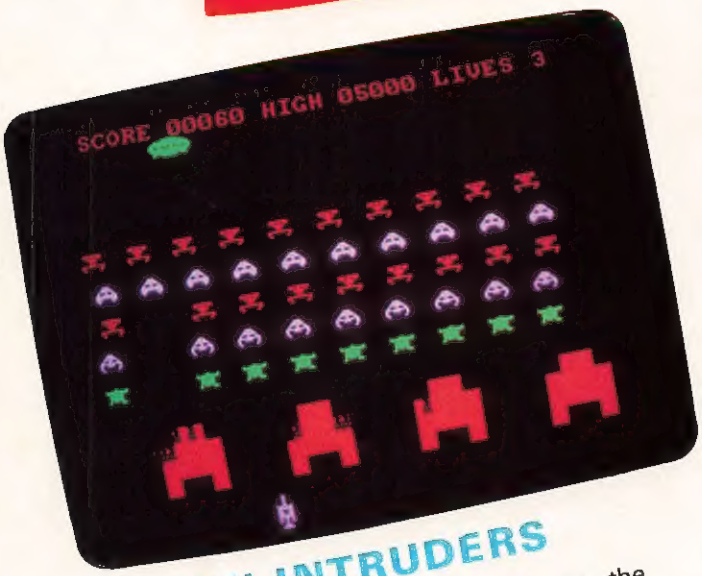
Subscription rates for 12 issues, post free:
£12 – UK
£15 – Eire (Sterling only)
£20 – Rest of world (surface)
£40 – Rest of world (airmail)

ABC Member of Audit Bureau of Circulations



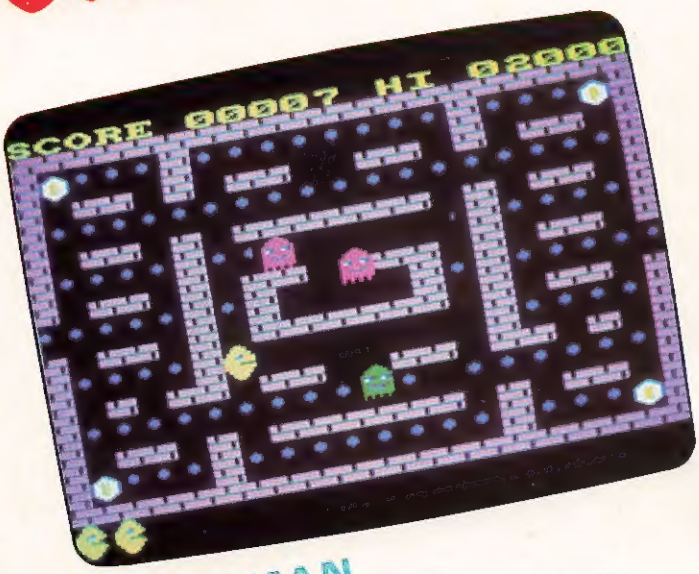
Contents

4 GREAT GAMES FOR THE PRICE OF ONE!



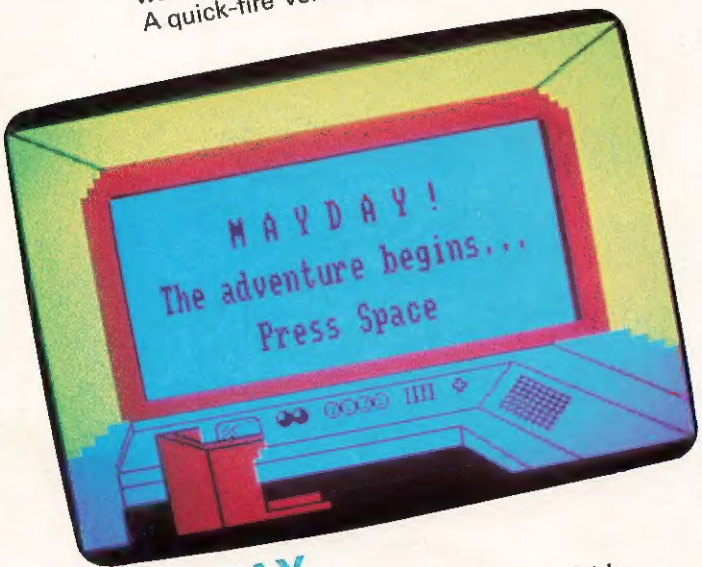
ALIEN INTRUDERS

With only your laser for protection, destroy the waves of aliens who threaten to engulf you. A quick-fire version of an all-time great!



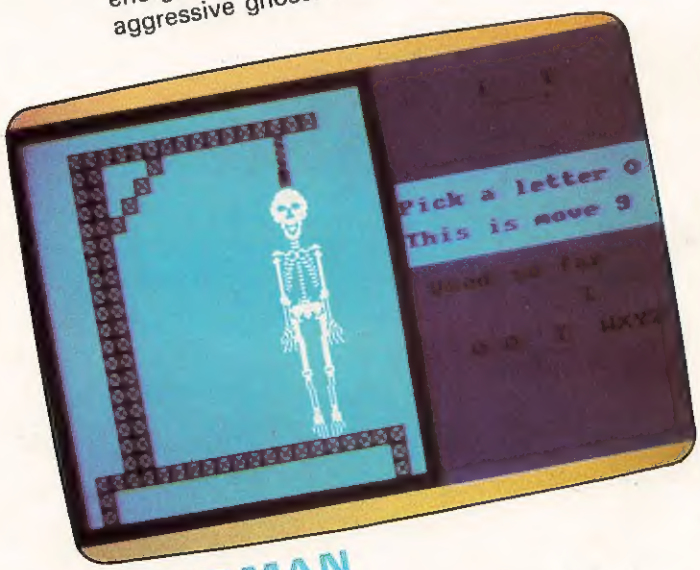
SNAPMAN

Steer your man around the maze, gobbling up energy pellets while keeping away from the aggressive ghosts who are out to get you.



MAYDAY

Guide the sole survivor of a stricken spaceship through the wreckage of his craft. Recover vital medical supplies . . . or a planet is doomed!



DEADMAN

The time honoured game of suspense – but with some novel endings. Guess the hidden letters or something nasty could happen to you.

Computing with the Amstrad presents

CLASSIC GAMES on the Amstrad

Here's something really special from *Computing with the Amstrad!* We've commissioned four rip-roaring programs that no games collection is complete without – the kind of games that really stand out in the short history of microcomputing.

This value-for-money package includes two top-rate machine code arcade classics plus a traditional word game and a futuristic adventure.

There's hours of enjoyment and something to suit everyone in this superb collection.

Please send me Classic Games on the Amstrad Vol. 1 (✓)

Cassette £5.95 *6048* ☐
3" Disc £9.95 *6049* ☐

Payment: please indicate method (✓)

☐ Access/Mastercard/Eurocard

No. ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐

☐ Barclaycard/Visa

No. ☐☐☐☐ ☐☐☐ ☐☐☐ ☐☐☐

☐ Cheque/PO made payable to Database Publications Ltd.

Name _____ Signed _____

Address _____

_____

_____

Send to:
***Computing with the Amstrad*, FREEPOST, Europa House, 68 Chester Road, Hazel Grove, Stockport SK7 5NY.**

(No stamp needed if posted in UK) *Please allow 28 days for delivery*

You can also order by phone **061-480 0171** *24 hours*

Don't forget to quote your credit card number and full address.

Now YOU can fly with the legendary Red Arrows – in the most challenging flight simulation ever!

It's the most exciting flight simulator ever written for a home computer – the product of many months of dedicated work by some of Britain's top programmers, enthusiastically aided by the talents of aircraft designers, engineers, mathematicians – and the Red Arrow pilots themselves.

Every ounce of power contained in the micro, and its enhanced sound and graphics capabilities, is used to give the utmost realism to re-creating the most spectacular aeronautical displays ever seen in the skies of Britain.

You start by practising take offs and landings. Then, once you have won your wings, you fly in formation as part of the Red Arrows team. There's no margin for error as you fly a mere six to 10 feet from each other – at speeds of between 300 and 350 miles an hour!

But the real drama begins as you plunge into the death-defying manoeuvres that have been thrilling crowds at air shows for the last 21 years.

On the panel in front of you are all the instruments you need – plus a screen giving you an external view of the complete formation you are flying. Slip out of line for a second and the eagle-eyed Red Leader will be on the radio ordering you back into position.

The program comes with a detailed flight handbook that will soon give you the confidence to take YOUR place alongside the ace pilots of the Red Arrows, even if you've never flown before!

Be a VIP visitor with the Red Arrows!

Everyone who buys a Red Arrows computer program will be invited to enter an exciting competition. The winners will be given a VIP visit to the Red Arrows base at RAF Scampton, the wartime home of the Dambusters. Your visit will include two nights' accommodation at a luxury hotel. And while you are at Scampton you will be invited to sit at the controls of a Hawk. There will even be a flypast of the Red Arrows in your honour!

Now on sale at:

BOOTS COMET Currys Dixons
Greens John Menzies RUMBELOWS spectrum
WHSMITH and other leading computer stores

ORDER FORM

	Tape (£8.95)	5¼" Disc (£11.95)	3" Disc (£12.95)	3½" Disc (£12.95)
Amstrad	☐	N/A	☐	N/A
Atari	☐	☐	N/A	N/A
BBC B	☐	☐	☐	N/A
Comm. 64	☐	☐	N/A	N/A
Electron	☐	N/A	N/A	☐
Spectrum	☐	N/A	N/A	☐

I wish to pay by:

☐ Access/Mastercard/Eurocard No.

☐ Barclaycard/Visa No.

☐ Cheque/PO made payable to Database Publications Ltd.

Name _____

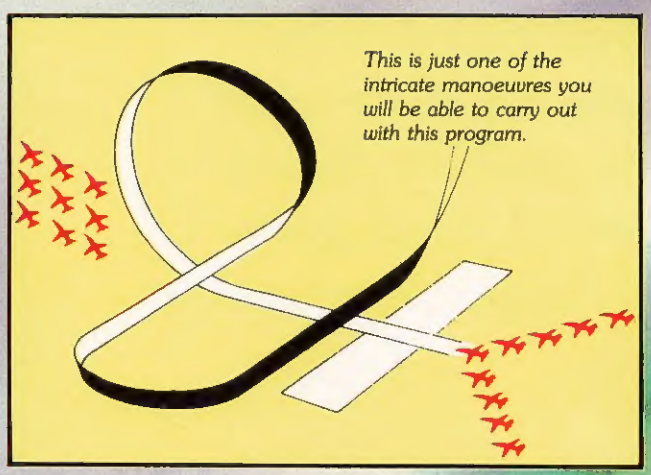
Address _____

_____

Signed _____

Send to: Database Software, FREEPOST, Europa House, 68 Chester Road, Hazel Grove, Stockport SK7 5NY.
(No stamp needed if posted in UK) *Please allow 28 days for delivery*

YOU CAN ALSO ORDER BY PHONE: **061-480 0173** *(24 hours)* *Don't forget to quote your credit card number and full address.* CA



This is just one of the intricate manoeuvres you will be able to carry out with this program.

Put yourself in the pilot's seat of the most manoeuvrable fighter in the RAF!


RED ARROWS
A gripping, realistic computer simulation for the
Commodore
Spectrum
Amstrad
Electron
BBC Micro
Atari

Let your fingers do the talking...with this special

Now you can teach your Amstrad to talk!

How it works

AT the heart of the dk'tronics speech synthesiser lies an incredibly powerful chip that has split the English language into its component parts – or allophones as they are known.

Altogether there are 59 allophones and five pauses stored in the speech chip's internal ROM. These can be combined to create a virtually unlimited vocabulary.

The potential of this chip is realised by dk'tronic's sophisticated, yet simple to use software. The brilliant program design enables the Amstrad to actually speak the words you type, in straightforward English, without having to resort to complicated phonetic spelling or difficult programming techniques.

Written to be as user friendly as possible, the synthesiser adds eight powerful commands to Amstrad Basic.

If you prefer complete control over your programs, though, full details are given for Basic and machine code programmers to exploit the tremendous scope of the synthesiser without using the software supplied.

In fact this system supports four different modes of use.

The first mode allows you to sound words using only the Amstrad's normal Basic commands. However, as you get more ambitious with your speech, a second mode is provided. This gives eight extra commands to use from Basic, making using the synthesiser even easier.

The third mode is the text to speech converter. When this is in operation speech can be typed in using normal English and the Amstrad does the rest. There's no need to work out the allophones as in the other two modes – the Amstrad does it for you.

As if all this wasn't enough there's the fourth mode. This has the synthesiser converting whatever appears on the screen into speech. Using this, you can literally listen to your listings!

YOU can add an exciting new dimension to computing with your Amstrad – with the help of this remarkable new product from dk'tronics.

It comes complete with the latest and very versatile speech chip, a powerful stereo amplifier and two high-quality 4in speakers, specially designed to match the Amstrad CPC464.

And because this is a special reader offer it comes to you at £5 off the normal retail price of of £39.95!

Fitting it is simplicity itself. All you have to do is to plug the synthesiser's interface into the floppy disc port at the back of the Amstrad and the jack plug into the stereo socket – and away you go!

With its volume and balance controls you will find you can put dramatic realism into the sound output of your Amstrad. All sounds that previously came from the Amstrad's 1½in mono speakers are now sent out via the interface in stereo.

So even when you're not using it as a speech synthesiser, it can bring startling depth and drama to the music and sound effects of all your favourite games!

These are the sounds – and pauses – you can create on your Amstrad

A	AE	26	fat	F	FF	40	fire	NG	NG	44	bans	TH	TH	29	thin
A	AY	20	great	G	GG2	61	go	O	O	23	cot	TH	DH1	18	they
AI	AIR	47	hair	IG	GG3	34	wig	O	OW	53	snow	TH	DH2	54	bathe
AR	AR	59	farm	GU	GG1	36	guest	O	UW1	22	do	U	AX	15	succeed
AU	AO	24	aught	GE	ZH	38	beige	OO	UW2	31	food	UH	OO	30	cook
B	BB1	28	rib	H	HH1	27	he	OR	OR	58	store	U	YY1	49	compute
B	BB2	63	big	H	HH2	57	hoe	OU	OW	32	ouch	V	VV	35	even
C	KK1	42	common	I	IH	12	fitting	OY	OY	5	toy	W	WW	46	wool
'C	C	8	uncle	I	Y	6	sky	P	PP	9	pub	WH	WH	48	whig
K	KK2	41	sky	IR	ER2	52	bird	R	RR1	14	real	Y	YY2	25	yes
CH	CH	50	church	J	JH	10	jury	R	RR2	39	brain	Z	ZZ	43	zoo
D	DD1	21	could	L	LL	45	luck	R	YR	60	fear	PA1	PA1	0	10 mS
D	DD2	33	do	L	EL	62	angle	S	SS	55	sat	PA2	PA2	1	30 mS
E	EH	7	bend	M	MM	16	milk	SH	SH	37	shirt	PA3	PA3	2	50 mS
E	EE	19	see	N	NN1	11	earn	T	TT1	17	its	PA4	PA4	3	100 mS
ER	ER	51	cater	N	NN2	56	no	T	TT2	13	top	PA5	PA5	5	200 mS

Column 1: Sound. Column 2: Allophone name. Column 3: Allophone number. Column 4: Example word.

Computing with the Amstrad reader offer!

Look at what this package offers you:

- ★ Speech synthesiser with almost unlimited vocabulary
- ★ Easy-to-use commands – it accepts normal English words
- ★ Built-in stereo amplifier with twin speakers
- ★ Programs can run while the speed chip talks

Eight additional Basic commands

|SPON	Speech on.
|SPOF	Speech off.
|FEED,n	Feed speech buffer direct.
|FLUS	Clear speech and text buffers.
|SPED,n	Speech speed.
|OUTM,1	PRINT text to speech.
|OUTM,2	Screen output to speech.
|OUTM,3	Output to screen and speech.

Please send me the dk'tronics speech synthesiser for my Amstrad CPC464

☐ I enclose cheque for £34.95 *(incl. VAT, p&p)* made payable to Database Publications Ltd.

I wish to pay by

☐ Access Card No. _____

☐ Visa Card No. _____

Signed _____

Name _____

Address _____

_____

POST TO: Speech Synthesiser Offer, Database Publications, 68 Chester Road, Hazel Grove, Stockport SK7 5NY.

CA6

Allow 28 days for delivery

MIRRORSOFT goes AMSTRAD!

Three of our best-selling children's programs are now available on cassette or disk for the CPC464.

There's fun and games with the Mr Men in:

FIRST STEPS with the Mr. Men

For children aged 4 and up, First Steps with the Mr Men introduces early directional and object and letter recognition skills in four colourful games. No reading skills are needed, and a key guide and full-colour instruction book/storybook accompany the program

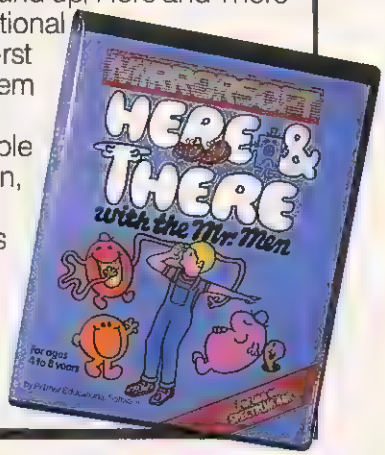
Cassette £8.95
Disk £11.95

and

HERE & THERE with the Mr. Men

For children aged 5 and up, Here and There takes the basic directional skills developed in First Steps, expanding them into actual verbal instructions and simple route planning. Again, there are four colourful games plus fully illustrated instruction book/storybook.

Cassette £7.95
Disk £10.95

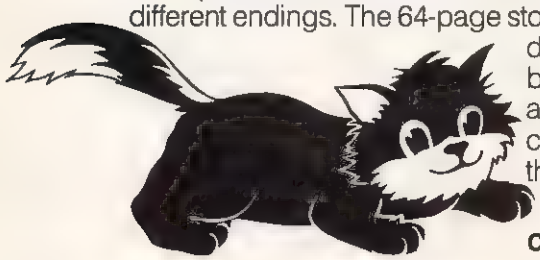


And bedtime stories will never be the same after

CAESAR'S TRAVELS

A unique interactive program/storybook package for children aged 3 and up, Caesar's Travels features 38 different story routes, with 18 different endings. The 64-page storybook expands the stories developed on screen, and both book and program can stand alone. Children will be captivated, and we think adults will find it a lot of fun too!

Cassette £7.95
Disk £10.95



And there's plenty more in the pipeline!

MIRRORSOFT programs are available from good software stockists everywhere or by direct mail from

MIRRORSOFT

Mirror Group Newspapers Ltd
Holborn Circus London EC1P 1DQ Tel: 01-822 3580

Amstrad's Alan Sugar with the new CPC664

Enter the 664

AMSTRAD'S second home computer – the CPC664 – has been warmly welcomed by the computer press.

It is essentially a CPC464 with built-in 3in disc drive and a number of enhancements to Basic and the operating system making the most of the integrated disc facility.

The machine comes with a choice of monitors and will cost £339 with the green screen and £449 with colour monitor.

Virtually all existing CPC464 software will run on the new 664 – plus a large proportion of the existing CP/M80 software base of 5,000 programs.

The CPC664 incorporates a built-in cassette interface so that existing tape based software can be loaded.

An Amstrad spokesman said: "We believe the CPC664 represents an extremely competitively priced package for the serious user upgrading from earlier machines.

"The first-time buyer now has an obvious choice of hardware that comes packed with the applications potential of CP/M and an existing base of more than 200 other items of software from the wealth of high-quality entertainment material available for the CPC464".

A second disc drive is also available for the CPC664, price £159.

SHARES SOAR

AMSTRAD boss Alan Sugar became darling of the City following the launch of the CPC664.

The company's shares, which had been suffering in the wake of the bad press that had hit Acorn and Sinclair, jumped 4p to 72p.

Talk in the City was that profits would be almost doubled when Amstrad's year ends on June 30. That would produce a cool £17.5 million.

Sugar himself refused to be drawn on profit forecasts. But he wasn't too reticent about sales of micros. He planned to sell 600,000 of them this year, he said – compared to 200,000 in 1984.

Coming soon:

Your own passport to the expanding, exciting world of communications!

IN response to considerable demand from our readers, *Computing with the Amstrad* is about to launch the first fully versatile communications package for the Amstrad.

Until now we have been unable to find a product that measured up to our own high standards in terms of value and performance. So we decided we owed it to our readers to make one available.

Fortunately we have been able to bring together a brilliant software team and one of the country's foremost modem manufacturers. In consultation with our technical experts, they have now come up with a communications package that represents the best possible value.

Our specifications were exacting. We needed a product that would encompass all the fascinating sources of electronic information available. These include:

★ ***Telecom Gold.*** *It's Britain's leading carrier of electronic mail. Coupled with MicroLink, the most exciting telecommunications innovation for years, it means that Amstrad owners will be able to communicate nationwide simply and cheaply with other enthusiasts without any of the restrictions of earlier systems.*

★ ***Prestel.*** *The source of 300,000 pages of information, exciting electronic magazines and clubs.*

★ ***Bulletin Boards.*** *Scores of these electronic notice boards are springing up all over the country, with a fascinating mix of news, views, helplines and programs.*

To accomplish all this, we needed a multi-baud rate, BT approved modem – and preferably one that was already well established. It was vital that it linked to the telephone network via a standard jackplug, avoiding the noise difficulties and signal loss often encountered with acoustically coupled modems.

And, of course, we had to have an extremely versatile piece of software, simple enough for the novice to use, but comprehensive enough to satisfy even the most demanding hacker.

Not only did it have to support all the above with the minimum of fuss, it also had to take non-standard systems and user-to-user communication in its stride.

As if all this wasn't enough, we wanted to be able to prepare messages off-line, then load them from tape or disc – and of course download to tape, disc or printer. And if we could log-on to MicroLink and Prestel automatically from disc...

Well we managed it. In fact we more than managed it, as you'll find out from the special communications supplement in next month's *Computing with the Amstrad.* In addition to revealing the final exciting details of our package, we'll be introducing you to the whole rapidly-expanding communications revolution, in our usual easy to follow manner.

Communications is the biggest thing in the micro world today and we're sure you'll want to be part of it.

Computing with the Amstrad will show you how.

Don't miss next month's issue – for an offer you cannot refuse!

Pictures are on tap

LATEST release for the CPC464 from software house Kuma Computers is Artwork, a graphics utility package enabling users to draw pictures.

It includes circle, ellipse, line and box drawing capabilities and allows full use of all available screen modes and colours.

Options for fast colour fill and complete screen scroll in all directions are included, plus the ability to load and save completed pictures to either tape or disc. Text may be included within pictures.

Artwork can be used to produce pictures for inclusion in other programs and has an extensive set of help screens accessible via the help menu, allowing even a novice to utilise the package to the fullest extent.

The software has been written to be fully compatible with Amstrad disc drives and interface. When discs are fitted the user can enjoy faster load and save times.

A demonstration picture is included on the tape to give the user an idea of the capabilities of the package, which costs £9.95.

SORRY, NO MARK II ROM FOR 464 OWNERS

THE arrival of the CPC664 has dashed the hopes of CPC464 owners wanting to upgrade their Basic.

The new Amstrad has the Mark II ROM, an extended 32k chip which contains an enhanced Basic which fills some of the gaps left by the old 464 version.

But the new ROM will *not* be made available to present owners of the CPC464, nor will it be going into the current crop of 464 machines being produced.

The reason given is compatibility with existing software from publishers who haven't followed the guidelines laid down by Amstrad for programming.

However observers feel that since the upward compatibility of the 464 with the 664 is much extolled by Amstrad, this reason is less than convincing.

At the launch Amstrad boss Alan Sugar intimated that "technical boffins" who contact Amsoft would be able to get hold of the Mark II ROM.

But when *Computing with the Amstrad* talked to Roland Perry, Amstrad's technical chief, the next day this was not the case.

A system of allocating ROMs would be introduced at an unspecified later date, but only to software houses and "genuine enthusiasts" was all Perry would promise.

Gorilla for Amstrad

MICRO POWER has released two arcade classics – one its best-selling Killer Gorilla – on a double-sided cassette for the Amstrad.

First released two years ago, Killer Gorilla is probably the biggest-selling game ever for Acorn Computers after Elite.

The other half of the package is Gauntlet, the graphics and action of which have been improved for the Amstrad.

A feature of the game is "stereo sound", which can be further enhanced by routing it through a hi-fi system. Price of the double-sided cassette is £9.95.

Getting all your sprites right...

A NEW utility product for the CPC464 has been released by Cable Software under the Electric Studio label.

The software, named Amsprite, contains a full sprite operating system and editor.

The fact that it is designed to be used from Basic and incorporated in the end user's own programs makes Amsprite one of the most powerful utilities on the market, says Cable.

The sprite operating system supports up to 40 sprites with all the usual assignable attributes including wrap, collision, bounce, obstacle, blank and reset.

The Sprite Editor/Assembler supports almost every conceivable function the user may require when designing his sprites, according to Cable.

The features include border colour section, view image, clear image, zoom image, partial or full scroll in four directions, flip, invert, mirror, animate selected images, load, file, store, recall, 16 colour mask, selection of pens with access to a 16-colour paint box.

More help for small firms

COPING with the day-to-day problems of running a small firm, or even a home, could be made easier thanks to the introduction of new business software for the Amstrad.

From Triptych, Dialog and Quest International, the programs will run on both the CPC464 and CPC664. They cover business aspects from invoicing to project planning and labelling of mail.

Triptych has three titles in its range – Entrepreneur, Project Planner and Decision Maker. They are available on tape, price £24.95 each, and disc, price £29.95. Entrepreneur is a business start-up kit and includes tutorial, teaching and application programs, and manual.

Project Planner is aimed at helping the small businessman plan his work more efficiently, allowing him to input his own jobs schedule.

Decision Maker has the same format but includes an 80 page manual, applications and teaching programs. It allows many "what if" considerations to be made during the decision-making process.

Dialog, currently enhancing all programs to make them fully disc compatible, has six titles.

The first, DFM Database and Mail Labels, is a comprehensive system with sort and search routine and a program to produce mail replies from database files. Price £14.95.

Other programs, which are all £29.95, include Day-base, a desk-top diary, Stockaid, a stock control system, Invostat to help with invoicing, the Transact book-keeping system and Home Accounts Manager.

Quest has taken its Padmede range of five business programs and adapted them for the Amstrad.

It has combined the three most useful to the small businessman – Sales Invoice, Sales Ledger and Stock Control a £99 price tag.

The remaining two programs, Produce Ledger and Nominal Ledger, are also being sold together at £66.

Mini Office lends a hand

MINI Office, the chart topping business package for the Amstrad from Database Software, has been officially endorsed as an aid for the handicapped.

The software is specifically recommended in "Micros for Handicapped Users", a book published by Helena Press of Whitby, Yorkshire. It carries a foreword by Baroness Masham of Ilton.

Revolutionarily priced at £5.95 – business packages can cost up to several hundred pounds – Mini Office is a suite of four programs.

All professionally written, they are made up of a word processor, database, spreadsheet and graphics.

The software package is singled out in the book in the chapter "Jobs for housebound people", which deals with the handicapped contemplating setting up their own businesses.

"We found Mini Office very useful", Peter Saunders of Helena Press told *Computing with the Amstrad.*

NEW BOOKS

THREE new books about the CPC464 have been published by Bernard Babani.

They are An Introduction to Z80 Machine Code (£2.25), An Introduction to Programming the Amstrad CPC464 (£2.25), and How to Write Amstrad CPC464 Games Programs (£2.50).

The publications are available from bookshops, computer stores and the larger branches of W.H. Smith.

Try flying – with the Red Arrows

THE world famous, death defying RAF aerobatics team the Red Arrows has helped create a completely new kind of flight simulator for the Amstrad.

Red Arrows, which is launched this month, has been written with the closest cooperation of the daredevil fliers themselves.

Programmers spent many hours with the pilots and their flight crews making sure the finished game would faithfully reproduce all the thrills of a real Red Arrows flying display.

The game puts the player at the controls of one of the nine British Aerospace Hawks in the Red Arrows team. It tests to the limit their skill and precision in a series of complicated aerobatic manoeuvres.

And each purchaser can enter a high score competition to win a weekend with the Red Arrows pilots at their base at RAF Scrampton, have lunch with them in their mess and sit at the controls of a Hawk. There will even be a Red Arrows flypast in their honour.

The game, which will be on sale at the many air shows where the Red Arrows are appearing this summer, costs £8.95 on cassette and £12.95 on 3in disc.

Part of the proceeds from the sale of the game will go to service charities, including the RAF Benevolent Fund.

Business accounts software

A COMPREHENSIVE suite of business accounting software packages for the CPC464 with CP/M and disc drive or the CPC664 has been released by Camsoft.

The new range includes Cambase, a database system, invoicing, stock control, sales, purchase and nominal ledgers, and payroll.

A wide range of integration levels is available and systems can run on single or dual drive computers. Price is £39 a module.

The range extends over 26 different combinations of modules to a fully integrated stock control, invoicing, sales, purchase and nominal ledger system available for £169. Telephone "hot line" support is included in the price.

The programs are written in machine code using the CP/M operating system and Camsoft's own CADO control system.

Updated Basic for new model

APART from the mechanical changes in the 664, a number of improvements have been made to the Basic 1.0 built into the CPC464. To mark these changes the Basic in the CPC664 is called Version 1.1.

First of the new features is a Fill command allowing an irregular shape to be filled with a colour.

There have always been facilities to change the graphics pen ink and graphics paper ink, also the plotting mode – or, xor, and. But two new commands, GRAPHIC PEN and GRAPHICS PAPER, plus additional parameters for PEN, MOVE, MOVER, PLOT, PLOTR, DRAW and DRAWR make these simpler to access.

For example, a MOVE command can set the graphics pen ink, and a PEN command can set the text background mode – transparent or opaque.

TAG now uses the current background mode giving the option of transparent paper. The command MASK is used to specify dotted lines.

New commands have been added to bring more of the firmware facilities into Basic.

COPYCHR$ is a new function that returns a character read from the screen and CLEAR INPUT is a single command to clear the keyboard buffer.

FRAME waits for frame flyback to occur, useful for the synchronisation of screen images, and CURSOR controls the display of the system cursor.

Several improvements have been made to Break handling and error recovery. On Break Cont will ignore a break caused by hitting Esc. All Escs can be trapped, including Breaks that occur in file operations.

A new function called DERR will return a disc error code, making it easy to handle a disc error from within a program.

The line editing and auto line numbering facility. AUTO have been enhanced.

Garbage collection speed has been improved and the passing of string parameters to external commands has been simplified.

Quiz taped

THE first of a computer game series modelled on the popular television quiz show 3-2-1 will be available to Amstrad users this summer.

Players will be able to send in their taped game plus scores and the top three will be invited to take part in a grand final. Price is expected to be £9.95.

More Amstrad CP/M

TIMATIC systems has started implementation of standard CP/M software packages for the Amstrad.

The company now has six business packages, nine CP/M languages and 12 system utilities available on Amstrad format 3in discs.

When your next assignment becomes a new label...

Sixth in MIKE BIBBY's helpful guide through the micro programming jungle

LET'S get to work right away this month! Take a look at Program I:

```
10 REM PROGRAM I
20 MODE 1
30 number = 0
40 WHILE -1
50 number = number + 1
60 PRINT "number is now ";number
70 WEND
```

Program I

It's identical to Program XII in last month's issue, and prints out the whole numbers from 1 upwards.

It's quite an interesting program, and well worth typing in – the next few programs are based on it.

Perhaps the most striking thing about Program I is the WHILE ... WEND loop. As we saw, WHILE and WEND act as "brackets" around a section (or body) of code we wish to repeat. In this case the lines we're repeating are 50 and 60.

Of course, you can't just have WHILE on its own – the micro needs to know WHILE what?

So we tag on to WHILE what is known as a condition – the "what" part.

Here the condition is simply –1. This, as we saw, is the micro's numeric code for true. So the condition is WHILE true, and since our micro believes everything until told differently, this boils down to WHILE forever.

In other words, we just keep on repeating the loop until we break out of it by pressing the Escape key twice.

Another interesting feature of the program is line 50:

```
50 number = number + 1
```

The effect of this line is to increase the value of *number* by one.

Remember, we do what's on the right of the equals sign first, so line 50 reads: *Take the value labelled number, add one to it, then give this new value the label number.*

The equals isn't really an equals sign as we know it – how could a number be equal to itself plus one? It's an *assignment* and you can read it as *becomes.*

Line 50 then reads: *Number becomes what was labelled number plus one.*

So line 50 increases the value of *number* by one and line 60 prints this new value out. Since these two lines are inside the indefinite WHILE ... WEND loop, the process keeps on repeating itself.

One last point to make is that, although I give *number* value zero in line 30, the first number printed out is 1. Can you see why?

Actually, line 30 is superfluous since the Amstrad assumes a numeric variable to be zero unless told otherwise.

Last month I challenged you to alter line 50 so that the numbers output went up in two's. Program II does the trick:

```
10 REM PROGRAM II
20 MODE 1
30 number = 0
40 WHILE -1
50 number = number + 2
60 PRINT "number is now ";number
70 WEND
```

Program II

Easy isn't it? I also asked you to see if you could make it increase in steps of 4 and 10. If you haven't managed it yet, I think you'll be able to now, using Program II as a guide.

Another poser I gave you was to start the output at 1000 and go down in steps of one. Here, instead of adding one in line 50, we'll have to take away one. Program III shows how it's done:

```
10 REM PROGRAM III
20 MODE 1
30 number = 1001
40 WHILE -1
50 number = number - 1
60 PRINT "number is now ";number
70 WEND
```

Program III

Notice that we give *number* the value 1001 in line 30. This is because we want the output to start at 1000. Before *number* is printed out (line 60), its value is decreased by one (line 50). By starting at 1001 we allow for this subtraction.

We also came across the first of an interesting tribe, the inequalities. Called in computerese "relational operators", they're concerned with which of two numbers is the greater or lesser, whether they are equal of not equal and so on.

You probably came across them at school. For example:

```
x > y
```

means that the value of x is greater

than the value of y.

Similarly,

```
x < y
```

means the value of x is less than y.

Of course, you tend to forget which way round the inequalities sign, >, should go. The way I remember it is to say that there are two ends – the wide or big end, and the pointed or small end. The "big" number goes next to the big end, the "small" number goes next to the small end.

This means that:

```
7 > 4
```

which reads *seven is greater than four* is true, while:

```
250 < 150
```

which reads *two hundred and fifty is less than one hundred and fifty*, is a downright lie!

```
10 REM PROGRAM IV
20 MODE 1
30 number = 0
40 WHILE number < 24
50 number = number + 1
60 PRINT "number is now ";number
70 WEND
```

Program IV

Program IV uses an inequality in the condition attached to the WHILE. Line 40 reads:

```
40 WHILE number < 24
```

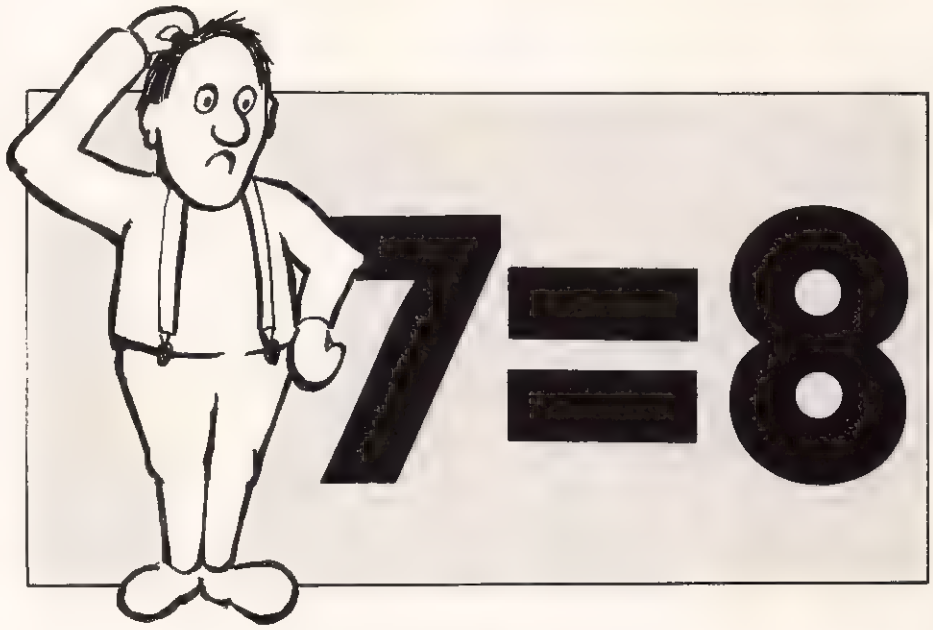
This means that we keep on repeating the loop bounded by the WHILE ... WEND so long as the value of *number* is less than 24.

As soon as line 40 meets a value of *number* that's not less than 24 it stops doing – *we say drops out of* – the loop.

That's straightforward enough, but something that might cause you concern is that the number 24 itself is printed out. You might think that, since the loop was to be repeated *while number was less than 24*, 24 itself shouldn't be printed out.

However, you're forgetting the effect of line 50 which adds one to *number* before printing it out. After the program has reached and printed out the value 23, the value of *number* is 23, so you are allowed to repeat the loop, since 23 is less than 24.

Line 50 then increases the value of *number* by one to give 24 and line 60 prints it out. Of course, since *number* is now 24, the loop won't continue.



Try running the program with line 40 changed to:

```
40 WHILE number > 24
```

Nothing seems to happen, does it? This is because, since line 30 sets *number* to zero, the condition for the loop (in line 40) is not met.

Therefore the loop isn't done – we skip or drop through it. And, since there isn't any more program, it simply ends.

The same line of reasoning should explain the results you get if you now change line 40 to:

```
40 WHILE number < 0
```

As well as inequalities, there is an equality, the = sign. We can combine them in expressions such as:

```
number <= 24
```

This reads *the value of number is less than OR equal to 24.*

Similarly,

```
number >= 24
```

reads *the value of number is greater than or equal to 24.*

Of course, we met a combination of inequalities last month – the <> pair, meaning not equal to.

To see how one such combination works in practice, take a look at Program V. Line 40 reads:

```
40 WHILE number <= 24
```

which I don't think you'll have any problem interpreting.

When you run the program you'll discover that this time 25 is the last number output.

```
10 REM PROGRAM V
20 MODE 1
30 number = 0
40 WHILE number <= 24
50 number = number + 1
60 PRINT "number is now ";number
70 WEND
```

Program V

You see this time, unlike Program IV, you can enter the loop with *number* equal to 24. Line 50 then increases it by one, and line 60 prints out the resulting 25.

When it tries to do the loop again, though, *number* isn't less than, or equal to, 24 and so it skips the loop.

The last time we tried to use greater than in the condition for a WHILE ... WEND loop – in a modification to Program IV – we didn't achieve anything too spectacular. Program VI shows how it can be useful in this role.

Before you run it, have a stab at working out what the last number printed will be.

```
10 REM PROGRAM VI
20 MODE 1
30 number = 51
40 WHILE number > 30
50 number = number - 1
60 PRINT "number is now ";number
70 WEND
```

Program VI

Well, from line 40 we know that the loop will only be done if the value

of *number* on entry is greater than 30. (Incidentally, line 30 ensures we start bigger than 30 – and line 50 tells us we're going down in ones.)

The last possible value that will "activate" the loop will be 31. It can't be 30 or under, since then *number* wouldn't be greater than 30.

So the last time the WHILE ... WEND loop is entered, *number* is 31. Line 50 then decreases it by one and line 60 prints out the resulting value, 30.

If the micro then tries to do the loop again, it won't be allowed to, since 30 isn't greater than 30.

WHILE ... WEND loops aren't the only Basic words we can use with conditions. There's the IF ... THEN statement as well.

IF ... THEN couldn't be easier to use, since it mirrors the English language so well.

For example, there's the standard parenting phrase:

```
IF you don't go to bed THEN
I will get cross
```

or the marital:

```
IF he says that once more
THEN I will explode
```

The idea is you give a condition after the IF, and after the THEN you specify the dire consequences if that condition is met.

Of course, if the condition isn't met, then you don't do what's after the THEN.

(Did you notice that last sentence is an IF ... THEN statement?)

Often the condition we specify in our programs will involve equalities or inequalities. For example:

```
IF number > 1000 THEN PRINT
"That's a big number"
```

or

```
IF number <> guess THEN
PRINT "You guessed wrongly"
```

Program VII uses these ideas to tell us whether a number is greater than, equal to or less than ten. We input the number in line 30, then lines 40, 50 and 60 screen it for each of the three cases or conditions.

If *number* matches one of the conditions (that is, it's greater than, equal to or less than 10), the appropriate message is printed out.

```
10 REM PROGRAM VII
20 MODE 1
30 INPUT " A number";number
40 IF number > 10 THEN PRINT "number
is greater than 10"
50 IF number = 10 THEN PRINT "number
equals 10"
60 IF number < 10 THEN PRINT "number
is less than 10"
```

Program VII

Notice that it's impossible for any value of *number* to cause more than one of the three messages to be printed out.

Program VIII develops the ideas we've been discussing. Lines 50 to 80 do the work of Program VII, comparing the number to 10.

```
10 REM PROGRAM VIII
20 MODE 1
30 counter = 0
40 WHILE counter < 3
50 INPUT "A number"; number
60 IF number > 10 THEN PRINT "number
is greater than 10"
70 IF number = 10 THEN PRINT "number
equals 10"
80 IF number < 10 THEN PRINT "number
is less than 10"
90 PRINT
100 counter = counter + 1
110 WEND
120 PRINT "Finished"
```

Program VIII

The new factor is that this comparison is wrapped up in a WHILE ... WEND loop. We want to do our comparisons three times (heaven knows why!). So I've introduced the appropriately named variable *counter* to keep track of how many times we do the WHILE ... WEND.

Initially *counter* is set to zero (line 30), and each time through the loop it's increased by one (line 100). So, the first time through the loop *counter* is zero, the second time it's one and the third time it's two.

Since we want the loop done three times only, we'd better make sure that *counter* never gets to three – otherwise it will have done the loop a fourth time.

We cater for this in the condition attached to our WHILE

```
40 WHILE counter < 3
```

It may seem a bit odd to have it less than three, but remember, we started counting at zero.

If you think it would be clearer, you could start counting at one by changing line 30 to:

```
30 counter = 1
```

You'll then need to change the loop condition by altering line 40 to:

```
40 WHILE counter <= 3
```

Incidentally, can you see how:

```
40 WHILE counter <> 4
```

and

```
40 WHILE counter < 4
```

are equivalent to this new version of line 40? Personally, I think our first alteration the clearer, since it specifies the 3.

Line 90, by the way, is just to space things out a bit, and line 120 lets you know the program has ended.

```
10 REM PROGRAM IX
20 MODE 1
30 WHILE answer$ <> "no"
40 INPUT "Shall we repeat this";
answer$
50 PRINT
60 WEND
```

Program IX

Don't forget that we can use the combination <> in the sense of "not equal to" with strings as well as numeric variables. Program IX shows a rather silly example of this.

It's very similar to a program we had last month, so it shouldn't cause you any problems.

```
10 REM PROGRAM X
20 MODE 1
30 IF answer$ = "no" THEN GOTO 70
40 INPUT "Shall we repeat this";
answer$
50 PRINT
60 GOTO 30
70 END
```

Program X

Now look at Program X. Believe it or not, this is the exact equivalent of Program IX. Instead of using WHILE ... WEND, though, I've used a combination of IF ... THEN and GOTOs.

Up till now, we've only used PRINT after a THEN. However, you can use any Basic keyword, including GOTO.

GOTO, which we've already met, makes the micro jump to the line number specified and carry on operating from there.

Line 40 inputs the value of a variable *answers$* in response to the question: "Shall we repeat this?"

Line 50 prints a blank line for space, then we jump immediately to line 30 via a GOTO in line 60.

Line 30 then analyses our response to the question. If we answered other than no, it continues with the (rather trivial) program. If we answered no, it leaves the program.

Line 30 uses a GOTO tagged on to the end of an IF ... THEN statement.

In effect it says, IF the value of *answer$* is no, THEN GOTO line 70.

So if we had input no, the effect of line 30 would be to make us GOTO line 70. Line 70 contains the Basic keyword END which, as its name suggests, causes the program to end execution.

On the other hand, any other answer doesn't meet the condition, so we don't do what's after the THEN, but simply continue with the program.

You might be wondering what happens the first time the micro meets line 30. After all, we haven't input any value for *answer$.*

Well, the Amstrad assumes that, until we've given a variable a value, it has no value whatsoever. It certainly can't have the value **no** then, so the program doesn't do what's after the THEN, but continues as normal.

I think you'll agree that this is an awful lot of work to achieve what we managed so simply with a WHILE ... WEND loop in Program IX, and the latter is far easier to understand.

GOTOs, with their tendency to leap about programs like March hares, are a prime cause of obscurity and hidden bugs in programs.

Good programmers tend to use them rarely – if ever – since there are often clearer and simpler alternatives. I hope you'll avoid them too.

● *Next month we'll look at another way to create loops, without using WHILE ... WENDs or GOTOs.*

Haystack Peripherals CPC464 Software Haystack Peripherals

8 MIDGROVE, DELPH, OLDHAM OL3 5EJ Tel: 045 77 5229

	RRP	OUR PRICE
A'n'F		
Chuckie Egg	7.90	6.90
ABACUS		
Cash Planner	12.95	11.45
Mailing List	17.95	15.95
Non Vat Accounts	17.95	15.95
Payroll	29.95	26.95
Purchase/Sales Ledger	29.95	26.95
Stock Control	17.95	15.95
Biology Revision*	12.95	11.95
Chemistry Revision*	12.95	11.95
History Revision*	12.95	11.95
Physics Revision*	12.95	11.95
ADDICTIVE		
Football Manager	7.95	6.95
Software Star	7.95	6.95
ALLIGATA		
Blagger	7.95	6.95
Defend or Die	7.95	6.95
ACTIVISION		
Ghostbusters	10.99	9.99
AMSOFT		
American Football	9.95	8.95
Amsgolf	8.95	7.95
Amsword (Advanced)	19.95	17.95
Amsword (Easy)	9.95	8.95
Astro Attack	8.95	7.95
Centrecourt	8.95	7.95
Concise Basic Spec.*	19.95	17.95
Concise Firmware Spec.*	19.95	17.95
Cubit	8.95	7.95
Decision Maker	21.95	19.95
Detective	8.95	7.95
Devpac Ass/Disass	24.95	21.95
Entrepreneur	21.95	19.95
Guide to Basic I	19.95	17.95
Guide to Basic II	19.95	17.95
Harrier Attack	8.95	7.95
Hisoft PASCAL*	34.95	29.95
Home Budget*	19.95	17.95
Hunchback	8.95	7.95
Jammin*	8.95	7.95
Jet Boot Jack	8.95	7.95
Master Chess	8.95	7.95
Masterfile	21.95	19.95
Mr Wongs Loopy Laundry	8.95	7.95
Mutant Monty	8.95	7.95
Pitman Typing Tutor	9.25	8.25
Project Planner*	21.95	19.95
Punchy	8.95	7.95
Quackajack*	8.95	7.95
Roland/Caves	8.95	7.95
Roland in Space*	8.95	7.95
Roland/Sq Bashing	8.95	7.95
Roland in Time	8.95	7.95
Screen Designer	14.95	12.95
Snooker	8.95	7.95
Splat!	8.95	7.95
Spreadsheet*	19.95	17.95
Starwatcher	17.45	15.45
Stockmarket*	8.95	7.95
Xanagrams	8.95	7.95
AMSOFT/BES		
Animal, Vegetable, Mineral	8.95	7.95
Happy Letters	8.95	7.95
Happy Numbers	8.95	7.95
Happy Writing	8.95	7.95
Map Rally	8.95	7.95
Osprey!	9.95	8.95
Timeman One	8.95	7.95
Timeman Two	8.95	7.95
Wordhang	8.95	7.95
Worldwise	8.95	7.95
ANIROG		
Flightpath 737	6.95	5.95
House of Usher	6.95	5.95
Moon Buggy*	7.95	6.95
Survivor	6.95	5.95
Zodiac*	6.95	5.95
3D Time Trek*	7.95	6.95
ARTIC		
World Cup Football	7.95	6.95
ASK		
Number Painter	8.95	7.95
BLACK KNIGHT		
Timebomb	7.65	6.65
CCS		
Warzone	6.95	5.95
CDS		
Castle Blackstar*	6.95	5.95
Steve Davis Snooker	7.95	6.95
C.P. SOFTWARE		
Bridge Player	9.95	8.95
Pinball Wizard	8.95	7.95
Superchess	9.95	8.95
CRL		
Rocky Horror Show*	8.95	7.95
CAMEL		
FlexiFrend	7.50	6.50
Grasp	8.50	7.50
DATACOM		
Empire	5.95	5.25
Execution	5.95	5.25
Intergalactic Trader	5.95	5.25
Snail Pace	5.95	5.25
Wumpus Mansion	5.95	5.25
DESIGN DESIGN		
Dark Star	7.95	6.95
Tank Busters	7.95	6.95
DIGITAL INTEGRATION		
Fighter Pilot	8.95	7.95
DURELL		
Combat Lynx	8.95	7.95
Death Pit	7.95	6.95
GEMINI		
Database	19.95	17.95
Home Accounts	19.95	17.95
Report Generator	19.95	17.95
GILSOFT		
The Quill	16.95	15.25
HEWSON		
Fantasia Diamond	7.95	6.95
Heathrow ATC	7.95	6.95
Technician Ted	7.95	6.95
HISOFT		
Font 464*	7.95	6.95
IMAGINE		
World Series Baseball*	8.95	7.95
INCENTIVE		
Confuzion	6.95	6.25
Millionaire	6.95	5.95
INTERCEPTOR		
Chopper Squad	6.00	5.25
Forest at Worlds End	6.00	5.25
Heroes of Karn	6.00	5.25
Jewels of Babylon	6.00	5.25
Message from Andromeda	6.00	5.25
KNIGHTSOFT		
Animated Strip Poker	8.95	7.95
KUMA		
Artwork	9.95	8.95
Database	14.95	12.95
Easi VAT*	39.50	34.50
Fruity Frank	8.95	5.95
Galaxie	5.95	5.25
Gems of Stradus	7.95	6.95
Holdfast	5.95	5.25
Home Budget	19.95	17.95
Logo*		TBA
Music Composer	9.95	8.95
Star Avenger	6.95	5.95
Zen Assembler*		TBA
LEVEL 9		
Adventure Quest	9.95	8.45
Colossal Adventure	9.95	8.45
Dungeon Adventure	9.95	8.45
Emerald Isle	6.95	5.95
Lords of Time	9.95	8.45
Return to Eden	9.95	8.45
Snowball	9.95	8.45
LOTHLORIEN (WARMASTER)		
Johnny Reb	6.95	5.95
Redcoats	6.95	5.95
Special Operations	6.95	5.95
MARTECH		
Brian Jack's Superstar Chall.*	7.95	6.95
MELBOURNE HOUSE		
Hobbit	14.95	12.95
Sherlock Holmes*	14.95	12.95
Sir Lancelot	5.95	5.45
MICROBYTE		
Erbert	6.95	6.45
3D Space Ranger*	7.95	6.95
MICROGEN		
Everyone's a Wally*	9.95	8.95
Pyjamarama	8.95	7.95
MICROPOWER		
Ghouls	6.95	6.15
Killer Gorilla/Gauntlet	9.95	8.95
MIRRORSOFT		
Caesar's Travels*	7.95	6.95
First Steps/Mr. Men	8.95	7.95
Here + There/Mr. Men	7.95	6.95
MOSAIC		
Erik The Viking	9.95	8.95
MYRRDIN		
Flight Simulation	11.95	10.45
NEW GENERATION		
Machine Code Tutor*	14.95	13.45
OCEAN		
Daley Thompson's Decathlon	8.95	7.95
Hunchback II	8.95	7.95
Kong Strikes Back	8.95	7.95
ODIN		
Nodes of Yesod*	9.95	8.95
P.S.S.		
Battle for Midway	9.95	8.95
R+B		
Mission One—Project Volcano	8.95	7.95
REALTIME		
3D Starstrike	6.95	5.95
REDSHIFT		
The Tripods	11.50	9.95
SOFTWARE PROJECTS		
Jet Set Willy	8.95	7.95
Manic Miner	8.95	7.95
SUPERSOFT		
Interdictor Pilot	17.95	15.95
TASKSET		
Superpipeline*	8.95	7.95
TASMAN		
TASCOPY	9.90	8.90
TASPRINT	9.90	8.90
TASWORD	19.95	17.95
TYNESOFT		
Supergran*	9.95	8.95
ULTIMATE		
Knightlore*	9.95	8.95
VIRGIN		
Sorcery	8.95	7.95
VORTEX		
Android One	7.95	6.95
WINTERSOFT		
Ring of Darkness	9.95	8.95

*** Please ring to confirm availability**

C15 Computer Cassettes (Box of 10) **4.50**

Pro-Ace Joystick **11.95**

PHONE FOR DETAILS OF OTHER SOFTWARE – WE CAN SUPPLY ALMOST ANY AMSTRAD TITLE AVAILABLE!

Overseas orders – Please add £1.00 per item.
Send sterling money orders

BEWARE OF CHEAP BOOTLEG SOFTWARE – ALL OUR TITLES ARE ORIGINALS!

Please Send:	Price
TOTAL	

ALL OUR PRICES INCLUDE VAT AND CARRIAGE

Cheques/PO's to:
HAYSTACK PERIPHERALS
8 Midgrove, Delph, Oldham OL3 5EJ
Tel: Saddleworth (04577) 5229

NAME .....

ADDRESS .....

.....

.....

Fly high in a jet fighter simulator

IF you've ever fancied your chances as a jet pilot, then Digital Integration's **Fighter Pilot** should appeal to you.

This aircraft simulation puts you in the pilot's seat of an USAF F15 Eagle jet fighter, with options to allow straightforward flying or air-to-air combat with enemy fighters.

From a menu of options you may choose to start your flight from take off position or landing approach. My disastrous attempts at landing soon convinced me that it was much easier to opt for take off.

Taking either of the combat options starts you off in mid-flight around 20,000 feet. You may also select combat practice, where you are positioned two miles behind the enemy plane at approximately the same height.

If you can keep tail on the enemy it will appear in your gunsights at a distance of one mile, at which point with a little bit of luck you can blast it out of the sky.

In practice mode the enemy doesn't fight back, but with some experience you may like to attempt true air-to-air combat.

With this option you will track the enemy with the help of your on-board computers before shooting it down.

Be warned though, the enemy returns your fire, and it becomes a real dogfight.

If you really want to live dangerously you can lob in crosswinds and air turbulence, not to mention a blind landing in foggy conditions.

The screen display is excellent. the top section is the pilot's cockpit view, in which you see the horizon, the runways on approach to landing, and of course the enemy if you are in combat mode.

The lower half is taken up by the instrument panels.

There is quite a lot of detail to digest here and one soon learns the importance of keeping an eye on the most important instruments.

Quite often I found myself carefully holding a correct course while paying no attention to the fact that my altitude was rapidly approaching zero.

The cockpit view can be changed to display a map of the area on which can be seen the four runways and various navigation beacons, etc.

Any enemy aircraft in the area are also identified if you are in combat mode. I must confess to being a flight simulator addict, and I was particularly pleased with this program.

The instructions are adequate, all the available options are described together with the instruments and controls, and there's a little technical information to help you gain some flying experience.

The program caters for keyboard or joystick control. Unfortunately I was only able to use the keyboard, which can be a little tricky at first.

I have no doubt that the use of a joystick would certainly improve one's control of the aircraft.

Geoff Turner

PLAYING SOCCER IN SAFETY

DISAPPOINTED with England's performances in international football? Think you could do better?

Well here is a chance to put your money where your mouth is. Get a few friends together and play Artic's **World Cup Football.**

The Amstrad World Cup can be played by between one and eight players. Should there be an odd number of competititors then some lucky person will have to play the computer.

At the start of the game each player chooses his team from a list of 10 international squads.

The computer makes the draw and displays the matches and the order in which they will be played.

Once the formalities have been dispensed with we can get down to the action.

The screen clears and is rapidly redrawn to display the centre section of the football pitch.

It is at this point that the music begins and the teams run out onto the pitch to take up their correct positions.

Each team has five players and the matches are played over 90 seconds.

During play only one of your players is active at any one time, indicated by a change in the colour of his shorts.

Tackling tends to be a matter of running into the chap with the ball and hoping to emerge from the crunch with the ball on your foot (just like the real thing).

Travelling up and down the pitch causes the screen to scroll from left to right. This scrolling is not what you would call fluid, but when you're busy hacking at someone's ankles who cares?

My favourite bit of the whole game is the bouncing action of the football when kicked. Clever use of shadow makes this very effective.

Control of your player can be achieved by various means. Two joysticks (expensive), one joystick and the cursor keys (unfair), and finally the cursor keys and a cluster of keys to the left of the keyboard.

I admit that control of such a game will always cause problems, but the provision of a user-defined key option would have made things much easier.

The Amstrad World Cup may have its one or two little niggles but these are outweighed by close attention to detail.

Should things continue to deteriorate in real life this could be the only match in town where you can take the family in complete safety.

Jon Revis

On a sticky wicket

CRICKET Captain, from Allanson Computing, allows you to try your hand as a county cricket captain without leaving your armchair. The program stores team lists for 17 counties, but the game itself allows only 16 teams to take part in the knockout competition.

Once you have decided which team to drop the program will display the names of players in all 16 teams and you have the option of changing as many players as you like.

This gives you the chance to join the Boycott controversy by dropping him from the Yorkshire team, or just being silly by fielding Maggie Thatcher, King Kong and Batman.

Before beginning the game you must choose the team you wish to captain and the difficulty level. If you have made team changes you can save the game at this point.

An unusual feature of the save game feature is that is saves a complete game rather than a data file. This means that the saved game can be reloaded and run without the original program having been loaded first.

There is a two player option and you must fight your way through the knockout competition with the computer controlling all the other teams.

Each innings lasts for up to 20 overs of 3 balls and the first round match starts with you being asked to press the keyboard when two boxes are flashing. Depending on which box is lit, you will either bat or field first.

The screen displays a three dimensional high resolution animated view of the field.

You are given information about the batting and bowling skills of your team members and must make decisions on batting order and bowlers.

Despite the good screen presentation and excellent sound effects, the game failed to give me a real sense of participation.

This is particularly apparent on batting, when the only decision you have to make is whether to chance making two runs or not. Other than that, all you can do is sit back and watch the game proceed.

With a little extra care this could have been an excellent game, but as it stands it failed to keep my interest for long.

Steve Lucas

Aliens from the arcades

DEFEND or Die from Alligata is the first Defender to be produced for the Amstrad. Since the original version hit the arcades it has maintained a cult following.

Its attraction lies in the fact that it is fast, furious, and incredibly difficult to play. But once you have mastered the undoubted skills required the satisfaction felt is worth every drop of midnight oil burnt.

In Defend or Die the marauding aliens are on the prowl again, and this time they are in the kidnapping business.

Flying down they pick up passing humanoids and whisk them off to Mars or wherever.

As the sole surviving star fighter pilot, your fellow man is relying upon you to save the day. Using your incredible skills you skim the planet's surface blasting everything that moves.

The screen acts as a window onto the planet's surface. Travelling left or right causes the screen to scroll smoothly in that direction, revealing more of the landscape.

A small radar display allows you to monitor alien activity elsewhere on the surface. It only displays a series of coloured dots but these are easily interpreted.

You can tell at a glance when some helpless humanoid is being dragged kicking and screaming skywards.

Even at these times all is not lost. Speeding quickly to the little man's aid you can unleash laser fire right between the alien's eyes.

With the alien destroyed the man begins to fall

Notes fine, but the chor

WISE men have lately taken to sternly telling us that the games era is over. The home micro is not a toy, they thunder, and its future depends upon the provision of serious and practical software.

Kuma has clearly been listening and offers **Music Composer,** "suitable for both expert and novice musicians". Since I softly play wrong notes on my electric organ at all hours, I qualify somewhere in between.

The 20 page manual is easy to follow although the two pages disclaiming liability for any damage worried me a mite.

After five minutes loading, my Amstrad exploded into Bach's Violin Concerto. It didn't sound like violins but was a fair rendition.

I followed prompts to obtain a display of the treble and bass clefs and an empty "top" and "bottom" stave – the lines you write musical notes on.

Coaxed by the manual I entered the musical scale from lower to upper C, played it back and experimented with alterations. This quickly familiarised me with all the functions.

Entering notes follows a simple logical sequence. Press M (middle octave) plus C (note) plus Enter plus Q and a quaver (half-length note) is printed at middle C.

Some musical knowledge did seem necessary, although you could enter notes at random and hear how they came out.

Rests and a selection of note durations from demi-semiquaver to dotted semi-breve, plus key signatures up to six flats or sharps are supported.

You work on sequences of up to 200 notes at a time and can join, copy or erase sequences, save them as data and load them back. The tempo – playback speed – has 20 settings and you can insert, delete or amend notes.

You can also hear, but not see, your masterpiece transposed into a different key.

It's an excellent idea, just not taken far enough. Much memory could be freed by substituting variables and VAL for the repetitive data numbers in the listing.

Machine code would free more and certainly improve the sometimes slow response.

There is a one note per position restriction and this overlooks the musician's need for chords on the base clef and criminally wastes the Amstrad's sound commands.

Display of transposition and dumping screen to printer as instant sheet music would have composers rushing to buy. And a "sound keyboard and print screen" option for the musically ignorant would be nice.

For off-computer use, you'll

earthwards. Flying with incredible precision you intercept him in mid air and return him safely to the ground, gaining 1,000 points in the process.

The sideways scrolling action is executed impeccably, with not a judder to be seen and the characters' shapes and colours are identical to the original version.

If all Alligata's future releases are up to this standard they will soon be the top manufacturer of quality Amstrad software.

Carol Barrow

Sorcery casts graphics spell

I'D like to introduce you to **Sorcery** from Virgin Games. It is possibly the best game yet to appear on any computer.

This arcade adventure for joystick only combines the complexity of an adventure with the skill of an arcade game. Not only that, it has the best graphics I have ever seen.

You play the last remaining sorcerer to escape capture by an evil wizard, the Necromancer. Your task is to roam the game's 40 screens using the objects you find to kill the demons that attack you on sight, and to free the other six sorcerers.

The screens are entered and left by small doors and you usually have several doors to choose from on each screen.

Any encounter with an enemy will reduce your energy, though you can top up by sitting on a cauldron!

Since you start from any one of five locations it is difficult to give specific hints.

However swords and similar weapons can be used to kill the hooded men who seem to spend most of their time cooking, and spells will kill the various demons and evil eyes that follow you.

Scrolls, bottles and a few other things will unlock doors or reveal hidden passages.

You score points for killing the Necromancer's minions and for freeing the sorcerers.

Overall, if this isn't the game of 1985 then I will be very surprised. Graphically it is a masterpiece. This is one game that is likely to sell out quickly!

Paul Gardner

d is lost

need to copy the screen by hand and add chords.

So, there's no expert appeal but the novice's long-suffering friends may find that the holiday slides torture is nothing compared to hours of "my latest composition".

Doreen Cox

FLEXI-FREND is a home budget program from Camel featuring forward planning, a built-in calculator, automatic logging and printer options.

I must admit that a program of this sort should be tested over a long period to discover its full potential or any drawbacks in the system.

Obviously this is not practicable for review purposes, but based on my experience and the potential described in the documentation, it certainly looks a promising program for the home market.

Computer home budgeting requires several essential features if the use of a computer is going to be more efficient than the traditional paper and pen methods of recording information. This program boasts many such features.

It can handle up to 35 different accounts which can be updated on a monthly cycle. The names of the accounts are determined by the user in the special option called Initial Set-up. Once established they will appear each time the program is used.

Changing the amounts in each of the 35 accounts can take place at any time during the month but the Monthly Salary option can only be completed once in each cycle.

Various updates such as standing orders are made automatically, and can only occur once a month.

The most attractive facility is the calculator. This can be called up at any time by pressing the Escape key twice, and is used like a conventional calculator.

It is useful for working out percentage interest and VAT, and the calculated amounts can be entered directly into the monthly records.

The printer options produce tables of information – another of the essential features of a successful home budget program.

However should a printer not be available all is not lost. The tables are easy to read from the screen, and balances and capital inputs can be displayed in histogram form.

The program is appealing because it has a straightforward display that it not cluttered with unnecessary information.

I would certainly recommend this program to the home user who wishes to set up his own easy-to-use budgeting system.

John Woollard

A home budget that looks ahead

Bovver for your hover

WHEN you are condemned to play computer games all night long it is great to receive a game as refreshingly simple and original as **Splat** by Incentive Software under the Amsoft label.

Unlike most arcade game heroes Zippy is not the space suited pilot of a sleek starship, but a runaway from a Flymo factory.

It is currently trapped in a maze, and as a benevolent

gardener you must guide our grass munching buddy through eight different levels.

If you are thinking at this point that there is nothing new about maze games, then you have never played Splat.

The maze in this game is much larger than the screen, which merely provides a window onto it.

Normally when this window technique is used it is the player who controls the scrolling direction of the screen. If you move upwards the screen also scrolls in this direction.

In Splat however the roles are reversed with the screen scrolling randomly in any direction. Using some very nimble footwork and a certain amount of anticipation you must prevent Zippy from becoming squashed against the wall, hence the title.

As Zippy explores the maze, an indicator at the bottom of the screen displays the percentage of the maze that he has covered.

Having explored 100 per cent he is guided to the next region of the maze and the indicator reset to zero.

Being a conscientious little Flymo, Zippy should be encouraged to take every opportunity to trim the tufts of grass that are found in the maze.

The scrolling, although supposedly random, can play some very dirty tricks. It is not unknown for the computer to tempt you with some succulent young shoots near the edge of the screen.

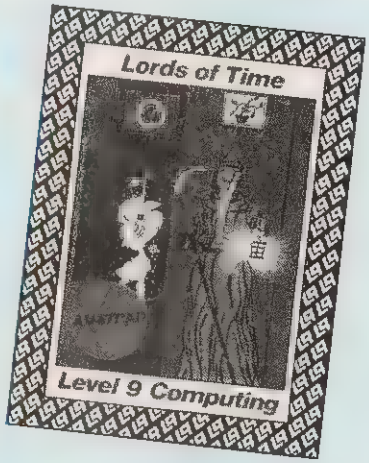
Beware, the chances are that the screen will decide to change direction just as you are in mid-mow and it's bye bye Zippy.

As you would expect, things get a little more tricky as you progress further. You can gain bonus points by collecting the plums, but you will find a lot more bovver for our hover in the shape of drawing pins.

Splat is an original and infuriatingly addictive game. You can't relax for one second without the screen suddenly changing direction and Splat! another one bites the compost heap.

Carol Barrow

Level 9 has a classic on its hands



ONE thing is always immediately apparent about Level 9 adventures – their excellence. **Lords of Time** is no exception.

The evil time lords are tinkering with Earth's history and must be stopped. You are chosen by Old Father Time to be the instrument of their downfall.

Your task – to collect nine objects, each inscribed with an hourglass, that are scattered through the ages.

You start in the present. A candle, some matches and an hour glass are all to hand and in the next room is a grandfather clock that is more than it seems.

After WINDing yourself up for the task ahead, you TURN to the problem at hand and PUSH forward.

The first time zone you visit is also set in the present. You'll find yourself TIED up a bit here.

Repeated contemplation of the compost heap on an empty stomach is no answer. However, you'll have to go to the ice age for the final solution to that coin.

Now you are prepared to tackle the second time zone. Here you'll find yourself with a mammoth problem. That can be the answer, Pirate Pete might say.

Further on in your quest you will frighten cavemen, fight Romans and wonder what pirates are doing with Vikings.

You will travel to medieval times, meet androids in the future and free imprisoned humans in the far future.

As usual with Level 9 adventures, there are the usual quota of puns.

This is especially worth remembering when you stand before the gate relecting on what has passed before. The program comes with a booklet giving information about playing the adventure and should definitely be read before starting.

It tells you to tackle the time zones in numerical order so that you are equipped to solve the problems in the next zone. This is true but you might also need to jump to other zones in order to get what you need to solve problems in the present one!

The adventure itself is not easy, Level 9 adventures never are. However it is easier than their Middle Earth trilogy.

All the puzzles are logical, if only in retrospect.

Overall, a superb adventure that is obviously destined to become a classic. Highly recommended.

Paul Gardener

Just one more hand...

I HAD no hesitation volunteering to review Knightsoft's latest release, as I fancied playing and improving my poker. I left the office on the Friday evening to a background of chuckles from the rest of the A team, and a chorus of "Yes, Al, we believe you old son. You go and

REVIEWED SO FAR

Adventure Quest – Level 9	Forest at World's End – Interceptor	Royal Quest – Timeslip
American Football – APS	Galaxia – Kuma	Screen Designer – DJL
Amstradraw – B.S McAlley	Gems of Stradus – Kuma	Snooker – CDS
Arnold goes to somewhere else – Nemesis	Ghouls – Micro Power	Snooker – Gem
Astro Attack – Amsoft	Happy Letters – B.E.S.	Snowball – Level 9
Amstrad Tutorial – Amsoft	Harrier Attack – Durrell	Software Star – Addictive
Atomsmasher – Romik	Happy Writing – B.E.S.	Sorcery – Virgin
Blagger – Alligata	Holdfast – Kuma	Splat – Incentive
Centre Court – Epicsoft	Home Budget – Kuma	Star Avenger – Kuma
Chopper Squad – Interceptor	Hunchback – Ocean	Strip Poker – Knightsoft
Colossal Adventure – Level 9	Lords of Time – Level 9	Survivor – Anirog
Country Cottages – Sterling	Map Rally – B.E.S	Test Match – CRL
Crystal Theft – Wiccasoft	Manic Miner – Software Projects	The Moors Challenge – Timeslip
Cubit – Mr Micro	Message from Andromeda – Interceptor	Timeman One – B.E.S.
Defend or Die – Alligata	Mr Wong's Loopy Laundry – Artic Computing	Timeman Two – B.E.S.
Detective – Argus Press	Music Composer – Kuma	Trial of Arnold Blackwood – Nemesis
Dragon's Gold – Romik	Mutant Monty – Artic Computing	War Zone – C.C.S.
'Er*bert – Microbyte	Number Painter – ASK	Wordhand – B.E.S.
Fighter Pilot – Digital Integration	Osprey – B.E.S.	Wordwise – B.E.S.
Flexi-fiend – Camel	Punchy – Mr Micro	World Cup Football – Artic
Flight Path 737 – Anirog	Pyjamarama – Microgen	Xanagrams – Postern
Football Manager – Addictive	Roland in the Caves – Amsoft	Z80 Assembler, Disassembler and Editor – Arnor
Fruit Machine – Amsoft	Roland Goes Square Bashing – Durell	
Fruity Frank – Kuma	Rollaball – Timeslip	

improve your poker".

Now the fact that this particular version is an animated **Strip Poker** really had no bearing at all on my choice, honest! And if you believe that, you'll believe there's life on the Moon.

The screen is laid out in three sections. Your hand is dealt and displayed in a column on the left. Your opponent, the delectable Mindy (a sort of Amstrad version of Victoria Principal), stands provocatively on the right, nearly wearing a slinky dress.

The centre is taken over by

Ossie, who runs the whole show. He's the croupier and takes the form of an animated playing card who, using an overhead speech bubble, tells you everything that is going on and offers all the alternatives during play.

These alternatives consist of the standard poker options – fold, call, raise and bet, and are selected by a single key press. The amount of your stake can also be raised and lowered in the same way.

Both you and Mindy start off with the same amount of money. The object of the exercise is to make her lose hers first and have to resort to gambling with one of her three items of clothing.

Now I'm not a particularly good poker player, but Mindy appears to play a fair game.

Even so, within an hour I had that dress off. The animation involved in this action was very clever and done in the best possible taste and must be seen to be believed.

"You can now play for my bra", says Mindy. "Yippee", shouts I, and we're off again. It took me nearly one and a half hours this time to have her stood there topless – with two strategically placed hands concealing her modesty.

An hour later found me betting my pants holding a full house of queens over sevens, intent on forcing her ladyship into a full frontal.

Alas her kings over tens had me beaten and I must admit in all honesty, that here I cheated a bit – I kept my pants on. She never knew of course, but the annoying thing was that she got her bra back – devious that, I thought, and shattering after nearly an afternoon's play.

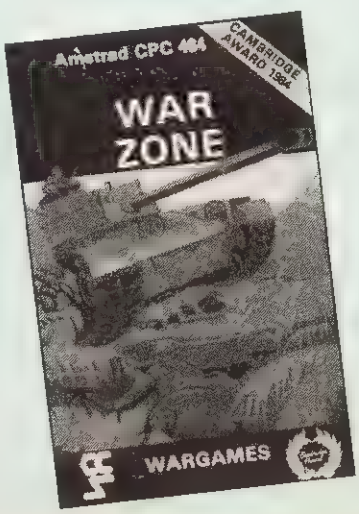
Undaunted I continued and after a couple of hours had Mindy in her birthday suit, feeling that I had just played an extremely good game of poker.

Whether Mindy plays a good game (she can certainly bluff) I really cannot say. I must leave that decision to someone who is a regular player.

I don't think this version is intended to be taken all that seriously. Even though it is saucy, there is nothing offensive in the graphics and my eight year old daughter sat pressing the keys for me at some stages during the session.

The game may not make a Maverick out of you, but if you like a bit of a gamble, with a touch of excitement and a bit of fun, then this will appeal.

Alan McLachlan



A little gem of a battle

THE last time I played a strategy game I found it over long, over slow and, since it required two players – the computer acting as referee and general dogsbody – not overworked either.

But that was long ago on an Apple in the days when the in thing was adventures and war games took a back seat.

I'm happy to report that **War Zone** by Cases Computer Simulations, has redressed the balance. It's busy, bright, brilliant. A little gem.

The battle is waged on a map divided into nine squares over terrain generated afresh for each game by the Amstrad. Protagonists are infantry, tanks and artillery.

Judging by their firepower the game is based on the '39-45 war; there is no evidence of nuclear weapons, and lasers are thankfully absent.

The player controls the Blue army, whose HQ is in the top left hand square, while the micro (Red, naturally) occupies bottom right. Only one square at a time is displayed, and if you have no troops in any particular square you are not allowed to see into it.

This permits the preparation of a rapid deployment force (jawohl, Herr General Rommel) which can swan over the nearest border and give the enemy a nasty shock.

At which point, you realise, the computer could gain considerable advantage by PEEKing (unfair comment, for the program is written in machine code, hence the speed) into its human adversary's borders. I can only say that I saw no great evidence of electronic naughtiness, (though I did lose rather a lot).

You are allowed from between 15 and 150 units at the start of a game, and I found it wise to select around 20 of each type, as the whole thing got rather cumbersome and time-consuming otherwise. That might merely mean that I'm more major than major general material.

Another option at the start is whether the computer should operate fast or slow. You appear to gain nothing by opting for the latter as you make your own moves in your own sweet time. I don't think computers benefit from second thoughts.

So, you've picked your troops and the micro has matched them. Now it takes a few seconds off to generate the terrain – at random – and place the pieces. Random again is the choice of who goes first.

If the computer wins here it will display the first square it is to operate in, and Red tanks, infantry et al will start advancing. Each piece is allowed a certain number of moves, modified by the countryside (faster on roads, bogged down in woodland) and one shot at any enemy in range.

Firefights are accompanied by realistic crashes and much flashing. The outcome of each encounter is displayed on the right hand side of the screen, plus other useful information and prompts.

Each piece is allowed one move and shot at the enemy (not necessarily in that order) per go. You can, for instance, order a tank to shell an enemy position and, if you survive the encounter, proceed to the top of a handy nearby hill ready for your next "round".

At this point it is worth noting that your tally of operational troops is displayed on the right of the screen, constantly downgraded as your men hit the dust, but the micro keeps its casualties to itself.

If you opt for a full scale battle and real time intervenes there is a save game option.

If you are a war games fanatic, buy it. If you are not, still buy it. You soon will be.

Peter Gee

Pair those registers for more micro power

Part VI of MIKE BIBBY's introduction to machine code

IF you thought we learned quite a lot last month, wait till you see what we've got lined up for you now!

Let's recap a little though. We've learned that the Z80 has several 8 bit registers – that is, internal memory locations that can hold one byte. We refer to them by the letters, H, L, B, C, D, E and A.

We've also met quite a few instructions involving these registers.

For example:

```
LD r,n
```

where r is one of the internal registers and n is a number in the range 0 to 255. This allows us to load a register with a number or constant.

Also:

```
LD r,r'
```

where r' is another of the registers. This allows us to transfer the contents of one register to another.

```
INC r
```

increases the contents of a register by one while:

```
DEC r
```

decreases the contents of a register by one.

So far these instructions work for all registers. We did find, however, that some demanded the use of the A register. For instance, we could peek and poke memory via the A register.

To poke we used:

```
LD (pq), A
```

and to peek:

```
LD A, (pq)
```

where pq is a two byte number specifying an address in memory. These two instructions are restricted to the A register only. There's no such instruction as:

```
LD B, (pq)
```

for instance.

A similar restriction applies to the ADD and SUB commands – they only apply to the A register.

They appear as:

```
ADD A, r
```

and

```
SUB r
```

(Notice you don't have to specify the A register for SUB. The micro knows you're subtracting r from A).

And you can also ADD or SUB constants to or from the A register.

So:

```
ADD A, n
SUB n
```

exist.

Lastly, we learned that increasing the contents of a single byte past 255 caused "the clock to start again".

That is:

255 + 1 => 0
255 + 2 => 1

Similarly:

0 – 1 => 255
0 – 2 => 254

We're really doing quite well with single byte registers. There's one difficulty though: the numbers are too small. After all, if we wanted to specify a part of the graphics screen, we could be dealing with numbers well over 255.

We're used to handling bigger numbers than can be held in a single byte every time we specify an address. All our addresses take two bytes, as we've seen. For example, &3000 consists of two bytes – a hi byte (&30), followed by a lo byte (&00). We've used this every time we've used CALL.

This is the instruction that's similar to a GOSUB – except we follow it with the start of a machine code address, not a line number.

The routine we call ClrText, which clears the text screen, starts at address &BB6C. Again, two bytes – hi byte &BB, lo byte &6C. When we translate the mnemonics into hex though, the lo byte comes before the hi byte:

```
CALL &BB6C
```

becomes:

```
CD 6C BB
```

So to handle things like addresses and numbers bigger than 255 we need two bytes. Single byte registers, therefore, tend to be a bit restrictive.

To overcome this, the Z80 allows us to pair our registers to give 16 bit registers:

H combines with L to give 16 bit register HL
B combines with C to give 16 bit register BC
D combines with E to give 16 bit register DE

This means we can do things such as LD HL, &AAFF.

This will load the register pair HL with the two byte, 16 bit number

```
address  hex code  mnemonics
 3000    21 FF AA  LD HL,&AAFF
 3003    7D        LD A,L
 3004    32 F8 2F  LD(&2FF8),A
 3007    7C        LD A,H
 3008    32 F9 2F  LD(&2FF9),A
 300B    C9        RET
```

Program I

&AAFF. H will contain the hi byte (&AA) and L will contain the lo byte (&FF). That's how they get their name: H for HIgh, and L for LOw.

Let's prove it with Program I.

So what are we up to? If you believe what I've said so far LD HL, &AAFF will store &AAFF in the register pair HL. The opcode for this is 21.

Notice that the value &AAFF, like an address, is translated into lo byte, hi byte form so LD HL, &AAFF translates as 21 FF AA.

To sort out which holds the hi byte, H or L, we then copy the contents of the L register into A with LD A, L and then poke the first byte of Hexer's "workspace" with LD (&2FF8), A.

To have a look at what's in the H register, we LD A, H and poke it into the next byte of workspace with LD (&2FF9), A.

Then, of course, there's our obligatory RET.

We're forced to do the transfers to A, by the way, because we can't poke the other registers into memory.

Incidentally, if you haven't already typed in our hex loader and simple monitor Hexer from our March 1985 issue, might I suggest you do so now. These articles have been written with it in mind.

If you run Program I and examine the workspace afterwards, you'll see that &2FF8 (where what was in L was stored) contains &FF, and &2FF9 (which holds what was in H) contains &AA.

So it's proved, a 16 bit load (that's what we call things like LD HL, &AAFF) stores the hi byte in H and the lo byte in L. Notice, by the way, I've arranged for the bytes to be stored in our workspace in the standard lo byte, hi byte form.

Program II should be familiar from last month. It puts an asterisk on the screen.

We could replace the instruction LD H and LD L (8 bit loads as they are

```
address  hex code  mnemonics
 3000    CD 6C BB  CALL ClrText
 3003    26 14     LD H,&14
 3005    2E 0C     LD L,&0C
 3007    CD 75 BB  CALL PosTCur
 300A    3E 2A     LD A,&2A
 300C    CD 5A BB  CALL CharOut
 300F    C9        RET
```

Program II

known) with a single 16 bit LD HL as in Program III.

We still want to load H with &14 and L with &0C. Remembering that the hi byte goes in H and the lo byte in L the instruction we want is:

```
LD HL, &140C
```

which translates as:

```
21 0C 14
```

in lo byte, hi byte fashion.

If you run it you'll see that Program III does in fact place the asterisk where we want it – column 20 (&14

```
address  hex code  mnemonics
 3000    CD 6C BB  CALL ClrText
 3003    21 0C 14  LD HL,&140C
 3006    CD 75 BB  CALL PosTCur
 3009    3E 2A     LD A,&2A
 300B    CD 5A BB  CALL CharOut
 300E    C9        RET
```

Program III

mnemonics	opcodes
LD BC,nn	01 n n
LD DE,nn	11 n n
LD HL,nn	21 n n

Table I: Opcodes for loading register pairs with constants

in the H register) row 12 (&0C in the L register).

We can load our other register pairs with constants, as you can see from Table I. Program IV uses a 16 bit load into the DE register pair as well as into the HL register pair.

It uses a routine called GrafLine (&BBF6) to draw a line on the graphics screen which we've previously cleared with ClrGraf (&BBDB). GrafLine uses register pair DE to specify the X coordinate, and register pair HL the Y coordinate, of the line's endpoint.

Its starting point is, of course, the last point visited by the graphics cursor, as in Basic's DRAW command. Since we've cleared the

```
address  hex code  mnemonics
 3000    CD DB BB  CALL ClrGraf
 3003    11 7F 02  LD DE,&27F
 3006    21 8F 01  LD HL,&18F
 3009    CD F6 BB  CALL GrafLine
 300C    C9        RET
```

Program IV

screen, this will be at (0,0).

As you'll see from the listing, our X coordinate is 639 (&27F) and our Y coordinate 399 (&18F), so GrafLine will draw a line from the bottom left to the top right corner of the screen. Run it and see.

The return to Hexer (or Ready message if you've called it from Basic) will cause the line to scroll off the screen. Can you alter Program IV so that it waits for a key before returning? You'll find the CharIn routine at &BB18 useful.

Now you've encountered GrafLine, why not use it to draw some pretty pictures? The routine corrupts all the registers we've met but the Amstrad's firmware, or operating system, keeps track of where the graphics cursor has reached, so this needn't worry you if you're simply plotting on from where you've just reached.

You might find the routine GrafMove at &BBC0 useful while you're at it. This is the machine code equivalent of Basic's MOVE, with DE specifying the X coordinate and HL the Y coordinate. Again, our registers are corrupted.

Incidentally if we'd loaded H, L, D and E separately our code would appear as in Program V.

```
address  hex code   mnemonics
  3000   CD DB BB   CALL ClrGraf
  3003   16 02      LD D,&02
  3005   1E 7F      LD E,&7F
  3007   26 01      LD H,&01
  3009   2E 0F      LD L,&0F
  300B   CD F6 BB   CALL GrafLine
  300E   C9         RET
```

Program V

Admittedly, the code is only two bytes longer in this version. However Program IV is easier to read, and by loading DE and HL directly conveys the idea of coordinates better. In machine code every improvement in clarity, no matter how slight, is an advantage.

Take a look at Program VI. It uses several routines we haven't encountered yet.

What we're doing here is to create a text window, then clear that window to PAPER 3, which is red, providing we're still in our default colours in Mode 1 (the way the Amstrad is at switch-on).

TextWin (&BB66) allows you to create a text window. H and D contain the two text column numbers that you want the window to be between, while L and E contain the rows. It also corrupts all our registers.

In the above case our window is between columns 10 (&0A) and 30

```
address  hex code   mnemonics
  3000   CD 6C BB   CALL ClrText
  3003   26 0A      LD H,&0A
  3005   16 1E      LD D,&1E
  3007   2E 0A      LD L,&0A
  3009   1E 10      LD E,&10
  300B   CD 66 BB   CALL TextWin
  300E   3E 03      LD A,&03
  3010   CD 96 BB   CALL SetPaper
  3013   CD 6C BB   CALL ClrText
  3016   C9         RET
```

Program VI

(&1E). and rows 10 (&0A) and 16 (&10). It will also be window #0, since we haven't specified any other stream.

We also use the routine SetPaper (&BB96) to set the text background colour. A must contain a legal ink number for the mode you're in, and both it and HL are corrupted by the routine.

Notice that we then call ClrText again to clear the window to the paper colour.

The Basic equivalent of the process would be:

```
10 CLS
20 WINDOW 10,30,10,16
30 PAPER 3
40 CLS
```

In Program VI I've loaded H,L and D,E separately – no 16 bit loads here! To prove just how good you are becoming at machine code, I want you to convert it so you load HL and DE with 16 bit loads. That is, get rid of those four single register loads.

And while you're at it, that window is an awkward size – when you use the Examine Code option in Hexer it won't quite fit will it? So you can adjust the window size as well.

So far we've concentrated on loading our register pairs with constants. Our instructions have been in the form:

```
LD rr,mn
```

where rr is the register pair and mn is the two byte constant to be loaded, m the hi byte and n the lo byte.

There are lots more things we can do with the register pairs, but for the moment we'll learn how to use them to peek and poke into memory.

So far we've done our peeking and poking into memory 8 bits at a time via the A register.

The instructions were:

```
LD A,(pq)
```

and

```
LD (pq),A
```

Notice the brackets around the pq. These show that we're concerned with the *contents* of the memory location specified, pq.

Using register pairs, however, we can manage to peek and poke two bytes, that's 16 bits at a time – ideal for handling big numbers and addresses.

To poke, or more accurately, load in to memory, we can use instructions of the form:

```
LD (pq), rr
```

where pq is the memory location we want to poke into, p being the hi byte and q the lo and rr is one of the register pairs BC, DE, HL.

Table II gives the opcodes. For the first time we're dealing with two byte opcodes, so together with the address, an instruction such as:

```
LD (&3000),BC
```

will take four bytes in all to specify.

mnemonics	opcodes
LD (pq),BC	ED 43 q p
LD (pq),DE	ED 53 q p
LD (pq),HL	ED 63 q p
LD (pq),HL	22 q p

Table II: Opcodes for 16 bit "pokes"

You might have seen that there are two versions of:

```
LD (pq),HL
```

one of which has a single byte opcode – that's the one we'll use for preference.

Fine, we know the opcodes, but what exactly do they do? Well, since we're loading *two bytes* into memory from the register pair, we'll naturally have to load them into two separate memory locations.

If the instructions were:

```
LD (&3000),HL
```

the effect would be to load &3000

'Register pairs are ideal for handling large numbers and addresses'

with the contents of L and &3001 (the next memory location up) with the contents of H.

Notice it goes into memory lo byte (L) followed by hi byte (H) – very convenient. To put it more formally,

```
LD (pq),rr
```

causes

```
(pq)   <== rr lo byte
(pq+1) <== rr hi byte
```

where the arrows mean *is loaded with*.

Think about it! pq+1 is the next memory location after pq. Both pq and pq+1 are in brackets since it's their *contents* we're changing, not the number pq or anything weird!

A bit of practical work should make this clear:

```
address  hex code  mnemonics
3000     21 34 12  LD HL,&1234
3003     22 F8 2F  LD (&2FF8),HL
3006     C9        RET
```

Program VII

Program VII simply loads register pair HL with &1234, and then pokes the contents of HL into the memory locations starting at &2FF8, Hexer's workspace.

Run the program and then examine memory from &2FF8. You should see that &34 (the contents of L) is stored in &2FF8 and &12 (the contents of H) is stored in &2FF9.

Ring the changes in the program by loading the other register pairs – BC,DE – with constants and then poking them into the workspace. As you'll see, poking with register pairs stores their contents in lo byte, hi byte order in two adjacent memory locations – the one specified and the next higher up in memory.

We can also peek memory using the register pairs:

```
LD HL,(&3000)
```

would place the contents of memory location &3000 in the L register and the contents of &3001 in the H register (notice the hi byte, lo byte order in memory yet again).

We have the brackets around the address to show that we're dealing with the *contents* of memory. After all, if we left them out we'd have:

```
LD HL,&3000
```

which is just a straightforward 16 bit load, which will load the *constant* &3000 into the HL registers. H will then contain &30 and L will contain &00.

```
LD HL,(&3000)
```

on the other hand is our peeking instruction. After this instruction, what's in H and L will depend on the *contents* of memory locations &3000 and &3001.

If you like, in Basic,

```
LD HL, &3000
```

would translate to:

```
L = &00
H = &30
```

whereas:

```
LD HL,(&3000)
```

would be:

```
L = PEEK (&3000)
H = PEEK (&3001)
```

Our general description of 16 bit "peeking" with register pairs BC, DE, HL is:

```
LD rr,(pq)
```

which causes:

```
rr lo <== (pq)
rr hi <== (pq+1)
```

Table III contains the opcodes. We've got four byte instructions again, save for the "duplicate" involving HL. The reason for the extra, shorter instruction involving HL is that this tends to be our favourite and most frequently used register pair, in much the same way as A is among our 8 bit registers.

```
mnemonics      opcodes
LD BC,(pq)     ED 4B q p
LD DE,(pq)     ED 5B q p
LD HL,(pq)     ED 5B q p
LD HL,(pq)     2A q p
```

Table III: 16 bit peeks with register pairs

At this stage in the game it is a little awkward giving you a decent example of a 16 bit peek – though believe me they're very useful instructions. Anyway Program VIII gives a rather contrived example.

```
address  hex code  mnemonics
3000     2A 00 30  LD HL,(&3000)
3003     22 F8 2F  LD (&2FF8),HL
3006     C9        RET
```

Program VIII

All this does is to copy the first two bytes of our program (which starts at &3000) into Hexer's workspace at &2FF8.

```
LD HL, (&3000)
```

puts the contents of &3000 into L, and the contents of &3001 into H.

Our next instruction:

```
LD (&2FF8), HL
```

is the poke instruction we've already met and puts the contents of HL into workspace, lo byte first.

When you now examine the workspace (from &2FF8 onwards), you should find the first two bytes are:

```
2A 00
```

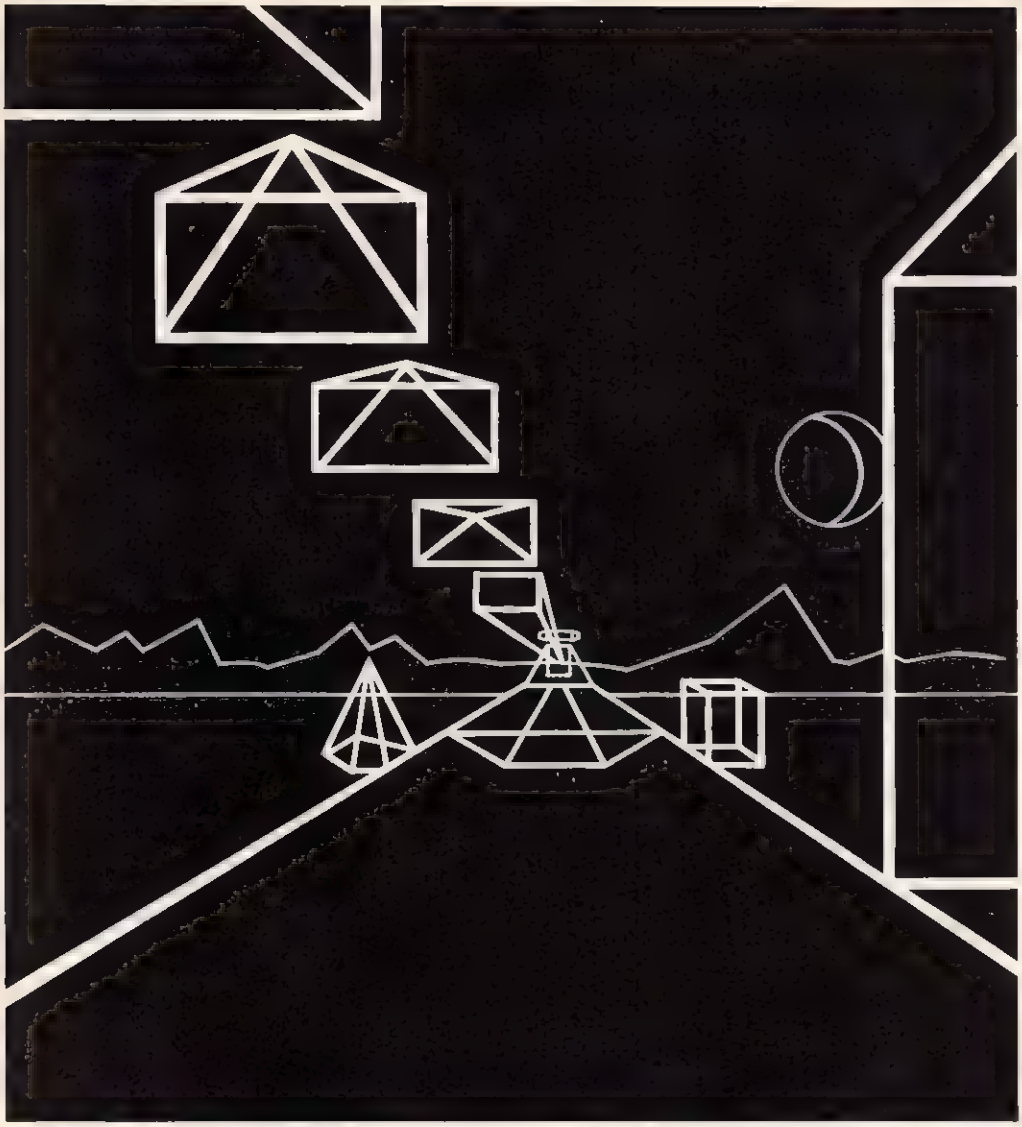
corresponding to the first two bytes of our program (stored at &3000 onwards).

Try rewriting the program using the other register pairs we've met to peek and poke with.

Well, that's plenty for this month. A final point – you can't combine any two single registers to give a register pair. There isn't a register pair HB, for instance, even if there's such a pencil!

And that word "combine" is important, too. The register pair HL, for example, isn't independent of changes you make to the single registers H and L. A change in either will affect the value of the two byte constant or address in HL.

● *Next month we'll see just how powerful our register pairs can be . . .*



TANK BUSTERS

Using the advanced graphics techniques developed for the game "Dark Star" the Design Design team bring you TANK BUSTERS. You take control of an advanced battletank on the battlefield of the future.
Your mission is to seek and destroy the enemy forces intent on your annihilation.
R.R.P. £7.95

DARK STAR

At last available on the Amstrad, the game that was awarded 10 out of 10 in Personal Computer News and the only game ever to receive a 100% rating in Crash Magazine. This is the fastest three dimensional space simulation available for any home computer.
R.R.P. £7.95

Design Design

Design-Design Software,
2 Ashton Way, East Herrington,
Sunderland SR3 3RX.

Trade enquiries to:—

125 Smedley Rd.,Cheetham Hill,
Manchester M8 7RS
Telephone 061 205 6603

SUPERCHARGE YOUR AMSTRAD.

'SUPERPOWER' SIDEWAYS ROMCARD FOR THE AMSTRAD

This unit opens up a whole new field of personal computing, previously only available to owners of the BBC Micro and other top of the range computers.

Rom-based software is instantly available from the keyboard and being written in machine code is very fast in operation.

The SUPERPOWER SIDEWAYS ROMCARD (Ref B101) has the following features:-

Matching Case, with easily detachable cover. Fits snugly to rear of computer.

Bus extension for fitting of Disk Interface or other units. Houses up to 7 foreground or background Roms – any mix of 8K and 16K.

Simple selection of programs from Keyboard – <BAR> Mail etc. No additional power supply necessary. Additional cards can be daisy-chained for further capacity.

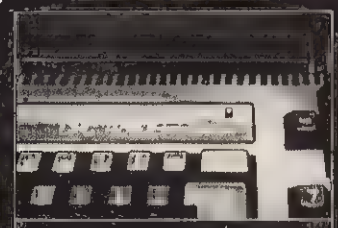
HIGH QUALITY BLANK EPROMS AVAILABLE

These are 200 nanosecond devices, the speed recommended by AMSOFT.

8K Eprom £7.95 (Ref B104). 16K £11.95 (Ref B105)

UNIQUE INTRODUCTORY OFFER

Until 30th June the SUPERPOWER ROMCARD will only be available by Mail Order. We are giving away a 16K Eprom (worth over £10) with all orders received by that date. Moreover, this will contain a FREE copy of Micro Power's highly acclaimed 'Ghouls' program to demonstrate the instant nature of Rom-based Software.

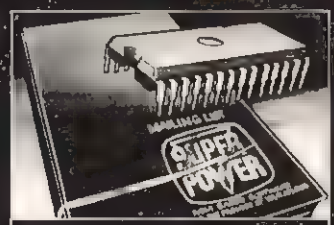


SUPERPOWER SIDEWAYS ROMCARD (Ref B101) £39.95.

'SUPERPOWER' ROM-BASED SOFTWARE

A complete range of Rom-based programs is under development, releases being planned at the rate of two per month, starting in June. Available from June 1st are the Mailing List/Club Membership program plus the Disk User's Utilities Rom.

Later releases will include Word Processor, Spreadsheet, Database and Graphics/Printer routines packages – with data interchange facilities – PLUS an Enhanced Basic/Toolkit program and a Machine-Code Monitor.



SUPERPOWER MAILING LIST and club membership PROGRAM (Ref B102) £39.95.

'SUPERPOWER' MAILING LIST (AND CLUB MEMBERSHIP) PROGRAM (REF B102)

The program handles very large lists of names and addresses on a selective basis, acts as a simple Database, and is particularly suitable for Club Membership records. The main features are as follows:- Each Record can contain up to 19 fields, those to appear on labels being user-selected. Variable length fields are used to optimise memory and disk space.

In practice, approximately 2000 records containing name and address and two non-label fields can be held on one side of a disk. Multiple double-sided disks are catered for. Each record can have up to 20 classification indicators. When used as Membership List, 12 can be nominated for monthly subscription reminders.

A screen report gives breakdown by categories. Printing options include Label fields only, and Total Record including classifications. Label can be of two standard sizes or user-defined. Program works with any parallel printer.

Alphabetical Order is dealt with on Entry. User choice of Keyword e.g. 'J' ohn or 'S' mith.

The Name field can be searched for the first part or the whole of a keyword. The whole file can also be searched for any string. There is sophisticated line and character editing, including change of keyword. Function Keys can be defined to give single key entry of commonly used string e.g. 'Membership No.' In Entry mode an automatically incrementing number is available. Foreground and Background colours can be selected.

SUPERPOWER DISK USER'S UTILITIES ROM (Ref B103) £39.95.

'SUPERPOWER' DISK USER'S UTILITIES ROM (REF B103)

This program allows detailed inspection and modification of information held on disk. It is of particular use in the recovery of data from corrupted disks. Individual sectors can be read from and written to. All data can be output to the screen and/or printer. The program also contains a number of functions of use to the assembly language programmer.

Main Functions.

FILELOAD – loads first sector into buffer and remainder into memory for fast access later.

READ – reads a sector into the buffer and enters Edit Mode.

EDIT – displays the current buffer. Data displayed is Buffer Address, Hexadecimal representation of each byte and ASCII representation of each byte.

Depending on mode selected, display is of 12 or 24 lines of 8 or 16 bytes.

- modification of Hex numbers, changing the ASCII automatically and vice versa.
- comprehensive cursor controls for easy editing.

FIND – can search a sector or total file for an ASCII string.

WRITE – writes a sector previously read by 'READ' or FILELOAD.

SUBSIDIARY FUNCTIONS

CATALOGUE – similar to AMSDOS catalogue.

MODE – Select 40 col./12 line display or 80/12, 40/24 or 80/24.

INK – Select Background and Foreground Colours.

ROM CHECK – lists all sideways Roms, giving Position, Foreground or Background, Name, Version No. etc.

OTHER DISK COMMANDS – Access to other commands such as Format and Verify is provided directly from the Rom.

ASSEMBLY PROGRAMMER'S AIDS. Disassembler. Relative Jump Calculator. Calculate the Sum and Difference of two hex numbers. Hex to Decimal Conversions – and vice versa. Intelligent Copy.

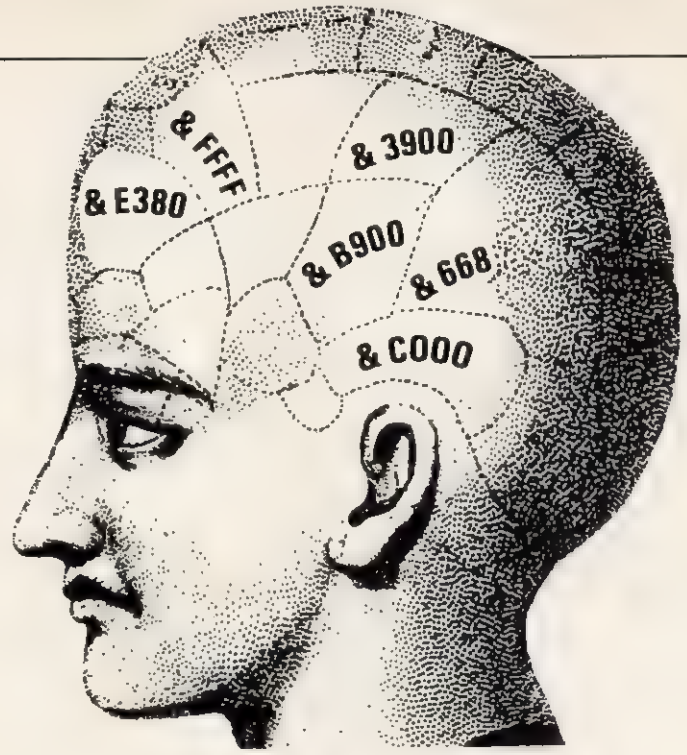
SUPER POWER

ROM-BASED SOFTWARE FULFILS THE PROMISE OF YOUR AMSTRAD

DEALERS! SuperPower represents an exceptional profit opportunity. Ensure your part of the action by becoming a SuperPower Advice Centre and Stockist and benefiting from our dealer support package. Contact Eileen Garfield on 0532 434006.

HOW TO ORDER Phone or write to: The SuperPower Project Manager, Sheepscar House, Sheepscar Street South, LEEDS LS7 1AD. TELEPHONE (0532) 434006. State your NAME and ADDRESS and the REFERENCE NUMBERS of the products you wish to purchase. Buy on Access/Visa card by stating your card number or write enclosing a cheque made payable to: Micro Power Ltd. (Please add 95p to your full order amount to cover post and packing.) ACT NOW TO SECURE YOUR ORDER AND YOUR INTRODUCTORY OFFER. (No cheques received will be banked before your order is despatched.)

KEEP UP TO DATE Phone (0532) 434006 to join our mailing list and receive regular bulletins on new products and release dates.



AMMON is a machine code monitor written in Basic. If you've used a monitor before on another micro you'll know how useful a tool it is. If not, you'll soon learn!

By using AMmon you'll be able to see how the CPC464 actually works – how variables are stored, the format of programs in memory and so on. You'll also be able to run your own simple machine code routines.

AMmon allows you to view any section of memory whether in ROM or RAM. Other commands allow you to edit memory and call machine code subroutines you've stored there.

The Amstrad has an unusual memory map. Firstly there's 64k of random access memory (RAM). In addition you can access another 32k of ROM memory, separated into two blocks of 16k.

The Z80 central processing unit (CPU) can only handle 64k of memory at one time. Since there is a total of 96k of memory in the basic machine the hardware must be designed so that the Z80 chip can access all of the memory present. It does this by switching sections of memory between RAM and ROM, more of which later.

From Basic the memory appears as 64k of RAM. But in machine code the memory can be RAM or ROM – selected by software commands.

ROM which "replaces" part of RAM is known as a background ROM. This is because it is "hidden" behind the RAM at the same location – it could be considered to be in the shadow of the RAM.

To access the operating system (OS) the lower 16k of memory must be switched from RAM to ROM – locations 0-&3FFF. The operating system is responsible for the overall running of the system and controls all of the input and output.

The Basic interpreter lives in the upper section of the memory map behind the screen memory – locations &C000-&FFFF.

This upper section of memory is "banked". Banking allows any one of 256 ROMs to be "placed" in the memory block &C000-&FFFF. (This 16k block of memory has been specifically reserved for ROMs).

Additional ROMs are banked in the same way as Basic. These can be used to allow new languages to be added to the existing system – another 252 such ROMs can be supported at present. The ROM at locations &C000-&FFFF is known as the upper ROM.

When any ROMs are enabled, write commands still access the RAM. This seems funny at first but when you consider the fact that ROMs are read only it then makes sense. The hardware handles this, so don't get too worried.

This leaves 32k of RAM between the two ROMs – assuming both ROMs are enabled. This is a "no man's land" and both ROMs use this as a workspace. Programs can also exist in this section of memory.

Now back to the monitor.

As we have no way of accessing either ROM from within a Basic program a machine code routine is required to fetch a byte from either ROM. Fortunately there are two simple calls which enable the upper and lower ROMs – turning off the RAM in the process.

These are called within a machine code program before the required byte is read from memory – so as to read the ROM contents instead of RAM. This routine resides in the "no man's land" at location &5000.

The routine to enable the upper ROM is at &B900. The lower ROM is enabled by calling &B906.

When you have typed in the program SAVE it before trying to RUN it. This is in case the machine code routine has been mis-typed – if this is the case the machine could crash.

RUNning the program displays the five available commands. Selecting the command is simply done by pressing the key corresponding to the command prefix. This will be D,E,J,L or P. *(A summary of the commands is given in Table I.)*

Selecting the D command will allow you to dump any section of memory. On pressing the D key you will be asked which memory type you wish to view. Option 1 will allow you to examine the upper and lower ROMs along with the RAM between them and Option 2 displays only the RAM.

After the memory type has been selected you will be asked for the start address for the dump. This must be in the range 0-65535. The number can be entered in decimal or hexadecimal – hex numbers should be prefixed with an & sign. Once the location has been entered an eight byte dump will be displayed on the screen.

The left hand column is the memory location in hexadecimal which is being dumped. The next eight bytes are the contents of eight locations starting from that location, again in hex.

On the right hand side of the display is the Ascii character of these locations' contents. For example, an A will be printed if a location contains &41 (65 in decimal). The memory will be dumped from the location specified until a key is pressed, or the end of memory is reached – this is at location &FFFF.

Memory master

Explore the Amstrad's hidden 96k memory with KEVIN EDWARDS

Next comes the E command which allows memory to be edited. This can be done in two ways.

The first allows you to edit byte by byte. The other option allows you to enter a string into memory. This is useful because it allows you to put messages into memory.

After E has been pressed the editing type is selected by pressing either 1 or 2. Selecting 1 allows you to edit memory byte by byte. Once selected, you will be asked to enter the start location you wish to edit.

Again, like all commands, the location can be entered in decimal or hexadecimal. Once this has been entered the memory location will be displayed on the screen along with its current contents. This is done first in hex and then in decimal.

You will now be prompted by a question mark to enter the location's new contents – a decimal or hex number between 0 and 255 (hex &0-&FF) should be entered.

After a byte has been entered the next location will be displayed in the manner previously described. Now this location can be edited.

Pressing Enter without any number returns you to the monitor's command mode. This should be used to exit the editing mode.

If you select editing Option 2 you are allowed to enter a string into memory. This simply puts the Ascii of each character of the string into memory in consecutive locations.

As before you will be asked to enter the start location – that is, where the string will be placed in memory. Once this has been done you will be asked to ent. the string. Type it in and press Enter.

The string will now be put in memory. If the string goes past the end of memory a message will indicate this and the function will be aborted. Be very careful with the editing command as it is a very powerful and "dangerous" feature of the monitor if abused.

The J command allows you to call a machine code routine. When selected you will be asked to enter the location to be CALLed. After you've done this you'll be asked if you wish to continue.

This is a sort of safety catch – calling a mis-typed address can cause havoc! Pressing N aborts the command while Y calls the sub-routine.

The L command simply returns back to Basic command mode.

The final command is P. This allows the Ascii contents of memory to be printed on the screen. When selected you will be asked to enter the memory type – ROM or RAM, as with the D command. Once this has been done you must enter the start location for the operation.

After it has been entered the start location is displayed in hex along with the Ascii contents of 32 locations starting from the base location given on the left.

As with the D command, the function can be exited by pressing any key. The command will also be terminated if the end of memory is reached.

All numbers displayed will be in hexadecimal unless otherwise stated. Any numbers entered can be in decimal or hex.

As you can see, the monitor offers some extremely useful commands. You may also wish to include some commands of your own. You'll find these can be added to the program without too much difficulty.

Now we've described what the program offers, let's see how it works in practice. Have a look at the following ROM locations using the D command:

&668 – The startup message along with various company names. Arnold was the "pet" name given to the Amstrad during software development.

&3900 – The character definitions starting from Space (each character definition takes up eight bytes).

&E380 – A list of Basic reserved words.

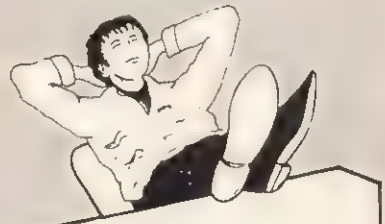
The Basic program text begins at RAM location &170 – reserved words are tokenised. Be very, very careful when editing between &170 and &1600 as you could destroy the monitor. Locations &5000-&500D contain the ROM read routine so don't edit this either.

Now you have a monitor you can explore the 96k of memory in the Amstrad. If you find some interesting locations write them down along with a description of what they are used for and send them to us. Happy hunting!

D – Dump memory in eight byte blocks.
E – Edit memory.
J – Jump to a machine code routine.
L – Leave monitor.
P – Print memory as characters.

Table I

```
100 REM Amstrad Monitor
110 REM (C)
120 REM By Kevin Edwards
130 ON ERROR GOTO 1180
140 MODE 1
150 PRINT"***  Amstrad monitor v1.0
***"
160 PRINT:PRINT"By Kevin Edwards"
170 POKE &5000,&CD:POKE &5001,0:POKE
&5002,&B9
180 POKE &5003,&CD:POKE &5004,6:POKE
&5005,&B9
190 POKE &5006,&3A
200 POKE &5009,&32:POKE &500A,&D:POKE
 &500B,&50
210 POKE &500C,&C9
220 PRINT:PRINT:PRINT"-- Commands --"
:PRINT
230 PRINT"<D>ump memory"
240 PRINT"<E>dit memory"
250 PRINT"<J>ump to subroutine"
260 PRINT"<L>eave monitor"
270 PRINT"<P>rint memory as character
s"
280 PRINT:PRINT:PRINT"Enter command .
..";
290 a$=INKEY$:IF a$="" THEN 290
300 a$=UPPER$(a$):PRINT A$:PRINT
310 IF INSTR("DEJLP",A$) THEN ON INST
R("DEJLP",A$) GOSUB 490,750,900,330,1
070 ELSE PRINT"Invalid selection !!":
GOTO 220
320 GOTO 280
330 END
340 REM SELECT START ADDRESS
350 PRINT:PRINT:PRINT"1...ROM &0000-&
3FFF, 0-16383      (0.S)"
360 PRINT"  & RAM &4000-&BFFF, 16384-
49151"
370 PRINT"  & ROM &C000-&FFFF, 49152-
65535  BASIC":PRINT
380 PRINT"2...RAM &0000-&FFFF, 0-6553
5"
390 PRINT:PRINT"Enter memory type, 1
or 2 ";
400 a$=INKEY$:IF a$="" OR (a$<>"1" AN
D a$<>"2") THEN 400
410 IF VAL(a$)=1 THEN rom=1 ELSE rom=
0
420 PRINT a$
430 PRINT:PRINT
440 INPUT"Enter start location ",star
t
450 IF START>65535 THEN PRINT:PRINT"E
nter a number between 0 and 65535 !!"
:PRINT:SOUND 1,200:GOTO 440
460 PRINT:base=start
470 IF base<0 THEN base=base+65536
480 RETURN
490 GOSUB 340
500 ZONE 5:PRINT HEX$(base),
510 FOR offset=0 TO 7
520 ZONE 3
530 location=base+offset
540 GOSUB 680
550 PRINT,HEX$(byte);
560 NEXT
570 PRINT,
580 FOR offset2=0 TO 7
590 location=base+offset2
600 GOSUB 680
610 IF byte>31 THEN PRINT CHR$(byte);
 ELSE PRINT CHR$(46);
620 NEXT
630 IF INKEY$<>"" THEN RETURN
640 base=base+8
650 IF base>65535 THEN RETURN
660 PRINT
670 GOTO 500
680 REM get byte from memory
690 IF rom=0 AND LOCATION<65536 THEN
byte=PEEK(location):RETURN
700 POKE &5007,((location/256)-INT(lo
cation/256)) *256
710 POKE &5008,INT(location/256)AND &
FF
720 CALL &5000
730 byte=PEEK(&500D)
740 RETURN
750 REM EDIT
760 PRINT:PRINT"1...Edit memory byte
by byte"
770 PRINT"2...Enter a string into mem
ory"
780 PRINT:PRINT"Enter type of editing
, 1 or 2  ";
790 type$=INKEY$:IF type$="" OR (type
$<>"1" AND type$<>"2") THEN 790
800 PRINT type$:PRINT:GOSUB 430
810 IF type$="2" THEN 900
820 ZONE 6
830 PRINT"&";HEX$(base),"&";HEX$(PEEK
(base)),base;"  ";PEEK(base),
840 INPUT newbyte$
850 IF newbyte$="" THEN PRINT:PRINT"E
nd of editing":RETURN
860 POKE base,VAL(newbyte$)
870 base=base+1
880 IF base>65535 THEN base=0
890 GOTO 830
900 PRINT:PRINT"Enter string to be pl
aced in memory"
910 INPUT".....>",a$
920 IF a$="" THEN PRINT:PRINT"Aborted
....!":RETURN
930 IF base+LEN(a$)-1>65535 THEN PRIN
T:PRINT"String over end of memory":RE
TURN
940 FOR loop=1 TO LEN(a$)
950 POKE base+loop-1,ASC(MID$(a$,loop
,1))
960 NEXT
970 PRINT:PRINT"Function complete..."
:RETURN
980 REM jump to subroutine
990 PRINT
1000 GOSUB 430
1010 PRINT:PRINT"Are you sure you wis
h to continue Y/N ";
1020 a$=INKEY$:a$=UPPER$(a$):IF a$=""
 OR (a$<>"Y" AND a$<>"N") THEN 1020
1030 PRINT a$:IF a$="N" THEN RETURN
1040 CALL base
1050 PRINT:PRINT:PRINT"Returned from
subroutine  ...!"
1060 RETURN
1070 REM print string
1080 GOSUB 340
1090 ZONE 5:PRINT HEX$(base),
1100 FOR offset=0 TO 31
1110 location=base+offset
1120 GOSUB 680
1130 IF byte>31 THEN PRINT CHR$(byte)
; ELSE PRINT CHR$(46);
1140 NEXT
1150 IF INKEY$<>"" THEN RETURN
1160 base=base+32:IF base>65535 THEN
RETURN
1170 PRINT:GOTO 1090
1180 SOUND 1,150:SOUND 2,100
1190 PRINT:PRINT:PRINT"ERROR, number
= ";ERR;" at line ";ERL
1200 END
```



Give your fingers a rest . . .

All the listings from this month's issue are available on cassette.

See our special offer on Page 77.

EXPANDING quadrilaterals

Analysed by Trevor Roberts

THIS month we're taking a look at an animation technique with expanding quadrilaterals. Easy to understand and simple to use, the program isn't limited to squares. Try changing the subroutine at 180 for different shapes.

And while you're at it, can you make the shapes come towards you?

```
10 REM expanding quadrilaterals
20 REM Trevor Roberts
30 MODE 0
40 square=1
50 xleft=119:ybottom=0
60 xright=519:ytop=399
70 GOSUB 180 draws first square
80 FOR square=2 TO 13
90 xleft=xleft+16:ybottom=ybottom+16
100 xright=xright-16:ytop=ytop-16
110 GOSUB 180 draws squares
120 NEXT square
130 GOSUB 250 lines now background co
lour
140 WHILE NOT true
150 GOSUB 300 selective switching
160 WEND
170 END
180 REM draw squares
190 MOVE xleft,ybottom
200 DRAW xright,ybottom,square
210 DRAW xright,ytop
220 DRAW xleft,ytop
230 DRAW xleft,ybottom
240 RETURN
250 REM colours to background
260 FOR hue=1 TO 13
270 INK hue,1
280 NEXT hue
290 RETURN
300 REM animation
310 FOR flash=1 TO 13
320 INK flash,24
330 FOR delay=1 TO 100:NEXT delay
340 INK flash,1
350 NEXT flash
360 RETURN
```

Draws 12 squares inside each other

initial square

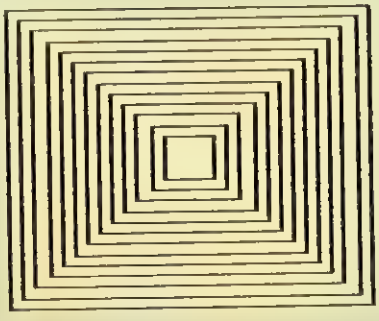
calculates corners of new squares

Endless loop calling animation routine

square drawing routine

Each pen in turn filled with ink of background colour

Changes ink in pen from blue to yellow then back to blue again.



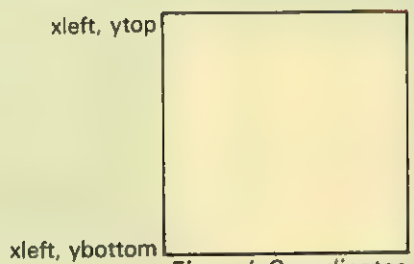


Figure I: Coordinates of square

Amstrad Analysis

30 Puts the Amstrad into Mode 0, the 16 colour mode.

40 Sets up the variable *square*, giving it the value 1. This will be used to keep track of the squares and the pens used to draw them.

50,60 These give the initial values to the variables *xleft, ybottom, xright, ytop.* These values are the coordinates of the corners of the outer square.

70 Calls the square drawing routine to draw the first square.

80-120 The FOR ... NEXT loop cycles 12 times, drawing a new square each time.

90,100 Calculate new values for the coordinates of the corners of the square. The offset means that each successive set of coordinates is inside the previous set.

110 Calls the square drawing subroutine using the new values.

130 This calls the subroutine which changes the inks in each of the pens to the background colour, blue. Effectively the lines all disappear.

140-160 These form an endless WHILE ... WEND loop. Each time it cycles the palette switching subroutine is called. It's these routines that provide the animation effect.

180-240 This subroutine is made up of the code that draws the squares. Figure I shows how the coordinates relate to the squares. The value of *square* in line 200 picks the pen which draws the lines.

250-290 The FOR ... NEXT loop cycles 13 times, changing the ink in pen *hue* to the background colour (ink 1).

300-360 These lines are responsible for the apparent movement of the squares. Each time round the FOR ... NEXT loop the ink in one of the pens is changed to yellow (ink 24), and then, after the delay of line 330, back to the background colour again.

The square appears briefly, then disappears again. As *flash* varies from 1 to 13, so each square in turn is shown. The result is animation.

8-PAGE PULL-OUT SPECIAL

EasyDraw

Your electronic brush and canvas!

EASYDRAW is a powerful graphics utility which you can use to create your own spectacular titles, pictures or games backgrounds. You can even build up a picture, save it part-finished to tape or disc, reload it at a later date and improve it until it suits your needs.

The picture you have created and saved can be used in your own programs either by loading straight to the screen or loading into the reserved memory for recall when needed.

The off-screen pictures shown on these pages took only minutes to produce, but of course it will take you longer until you become familiar with the facilities available.

A brief resumé of the capabilities of each option is shown in Table I.

You would be well advised to read the instructions in stages. There are 19 options and each needs practice to understand it fully and become proficient with its use. Read the guide to EasyDraw's functions down to the Beam option first. Then try drawing different coloured horizontal, vertical and diagonal lines. Once you understand these first five functions, build upon your knowledge by reading each subsequent option and practising it until you understand them all.

The following tips on drawing circles and polygons will help to make life a little easier for the first time user:

- Estimating the start position for plotting circles and polygons can present some problems. In the early stages these calculations can be made much easier with the use of a visual aid.

Draw a 100mm diameter circle on a piece of clear plastic using a compass and a felt tip pen. Then mark the circle as in Figure I. When estimating the start position for a plot, hold the plastic with the 0 and .5 vertical near the screen and read off the position required.

Some interesting shapes can be obtained quite simply by using the part shape and directional plot options as shown in Figures II and III.

- You can change the colour while plotting to produce a multicoloured shape. Choose a high number of sides to give you more time (there is no limit). A polygon with 1,000 sides plots slowly and gives you time to change colours or stop the plot wherever you wish.
- Draw to Tab can be implemented when drawing circles and will produce a cone shape, whereas setting Tab Move before entering the circle option and then Drawing to Tab will produce cylindrical shapes.
- If you make a mistake when

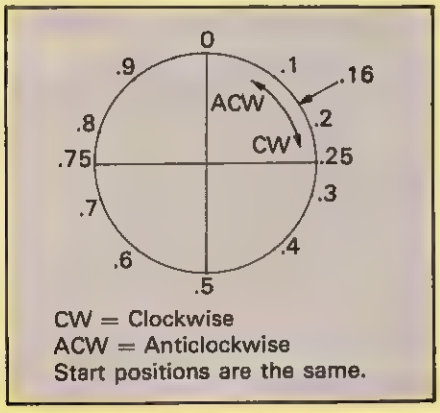


Figure I

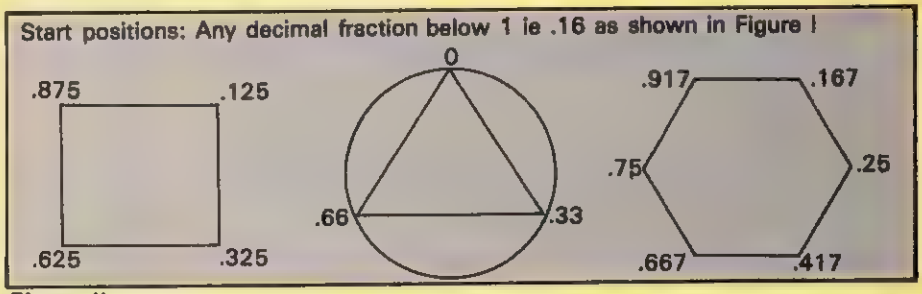


Figure II

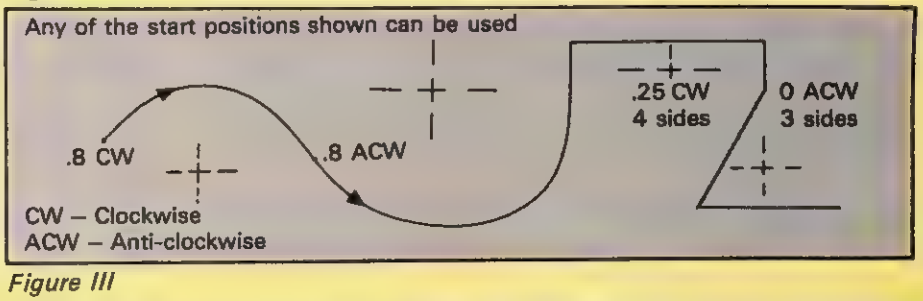


Figure III

drawing a shape you can erase it immediately by selecting the background colour and repeating the process you have just used to create the shape. However it can be difficult to remember exactly the sizes and positions used. A much easier way is to set Tab at one side of the shape, turn on the Draw to Tab function and as the cursor is moved the shape will be erased.

● To save a shape to memory you can draw directly to the screen freehand. Alternatively you can prepare a grid (Draw to Tab + Tab Move) to guide you. Change the drawing colour, set Tab, move to the next tab setting and draw a line to the previous tab then set the next Tab. In this way the shape is "memorised" while it is being drawn. Draw it at about half the size of the screen for the best results. A large shape reduces far better than a small one enlarges.

● To exit from EasyDraw press Ctrl+? as if you were going to save the screen to memory, but answer N to the first prompt and the option to End will be given. Escape can only be used in the main program when an input is required – such as Quick Circle – so this is the ideal way to access your listing to search for typing errors.

To run the program without the

User's Guide to EasyDraw's functions

Colour choice

To change the drawing colour press Shift + the corresponding letter key opposite the colour block.

Ink change

Press Ctrl + P followed by the appropriate letter key and Enter. The selected colour will turn black, at which point pressing Shift will move through the colour palette and Enter will fix the chosen colour.

Cursor control

The arrow keys control the cursor, and coupled with Shift will jump the cursor 20 pixels for faster movement.

Cursor jump

Ctrl + J will alter the size. Enter the horizontal and vertical size.

Beam

Ctrl + B toggles Beam on/off to draw lines, and cursor jump can be implemented. Alternatively Ctrl + Copy toggles the Copy key to draw but only when it is held down. With the latter option cursor jump cannot be implemented to draw – only to move.

Tab setting

The Tab key "sets" the cursor position which is memorised and indicated in yellow above the permanently displayed X, Y coordinates of the current cursor.

Lines can be drawn to Tab from any position, either on or off the screen.

Draw single line to Tab

Shift + Tab draws a line from the current cursor position to the previously set Tab position.

Draw to Tab

Ctrl + D toggles this option to draw a line to Tab with every move of the cursor – including any cursor jump implementation. This is useful for drawing several lines to Tab and avoids having to press Shift + Tab for each line as in draw single line.

Fill

Place the cursor inside any shape and Ctrl + F will fill it to the nearest vertical line under arrow key control as the cursor is moved vertically.

Erase

Selecting background colour A with the Beam, Draw to Tab, or Fill options will remove foreground colours. For full screen erasure press Shift + Clr.

Quick circle

Use the arrow keys to place the centre, press Ctrl + Q and either enter a radius > 4 or press 0. The latter enables the left or right arrow keys to move a second cursor with which the radius can be set by pressing Enter.

Circles, ellipses and polygons

Use the arrow keys to position the shape, then press Ctrl + Clr and input both horizontal and vertical radii using the same options as in quick circle.

Input the start position, a number between 0 and 1 (.5 will start to draw from the bottom of the shape). This is calculated using decimal fractions on a clockwise circle.

Input the number of sides. If this number is less than 20 you will be prompted for a delay. This is only needed if a part shape is required so that pressing the spacebar can halt the plotting.

Answer 0 or 1 to the prompt for clockwise or anti-clockwise plotting.

Tab move

The Tab setting can be made to move with the cursor by pressing Ctrl + Tab. Coupled with Draw to Tab parallel lines can be drawn using cursor jump to create a grid. This also works in the circle /polygon routine but must be set before Ctrl + Clr.

Shape memory

Ctrl + M stores and numbers upto 40 shapes to be recalled and drawn at any size. Press Tab, and while drawing a continuous line press Tab again at every change in direction (only Tab settings are memorised). Check your shapes using the redraw option before you save using Ctrl + S.

Ctrl + L will load a new shape file from memory.

Redraw shape

Ctrl + R and enter shape number and magnitude required (from .1 upwards), to enlarge or reduce the original.

Type at cursor

Press Ctrl + T to enter text, numbers or Ctrl graphics at the cursor position.

Go back to options

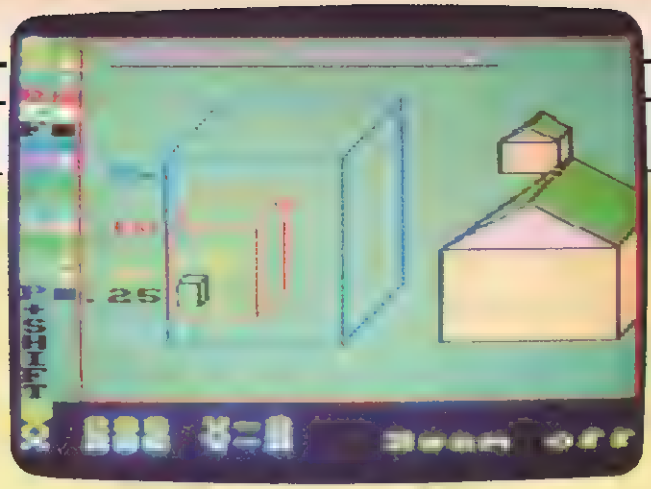
Ctrl + G will return you to the opening options for a mode change or to load a picture from memory.

Save screen

Ctrl + ? saves the screen to tape or disc.

Load screen

A saved screen can be loaded into memory using either the option at the start of the program, or Ctrl + L at any time.



opening screen use RUN 200. If the program stops as a result of a typing error and you are halfway through your first masterpiece don't panic! Put on the kettle, enter GOTO 550 the main program, and save the picture before you do anything else.

- Use a light touch in the Memory option as a prolonged keypress toggles the memory on and off quickly and will result in empty shapes being numbered in memory.
- Before saving any picture remove the flashing cursor by either moving it off the screen or by drawing a line in the colour of that position.
- When changing INKs make a note of the command given at the time as these are needed to be set up in your own program. Avoid changing INK 15 in (P):INK 4 in (E) and INK 1 in (B) as these are used for screen messages and you may end up with a one you can't see. Change them once your drawing is finished to any INK you wish.
- Several options use the same line for messages, so remember to switch them off when you have finished with them if you are using two or three at once.
- You can incorporate pictures created with Easydraw in your own programs. When a picture is saved the whole screen, not just the window, is saved. The border around the window is, however, saved in the background colour. So if you recall a screen in one of your games the border will be available for on-screen prompts.

A saved picture can be loaded directly to screen when, of course, it will be displayed immediately. Listing II shows the idea.

Alternatively you can load the picture into memory that's not screen memory. When you want to display it a simple machine code routine will move it from storage into screen memory.

This has the advantage that you can move your stored picture onto the screen, manipulate it, and resave it, as in Listing III.

SUMMARY OF OPTIONS

Beam	Ctrl+B
Cursor	Arrows
Line to Tab	Shift+Tab
Clear screen	Shift+Clr
Circle/Polygon	Ctrl+Clr
Draw to Tab	Ctrl+D
Fill	Ctrl+F
Go back to options	Ctrl+G
Input shape file	Ctrl+I
Jump cursor	Ctrl+J
Load screen	Ctrl+L
Memory shape	Ctrl+M
Pen ink change	Ctrl+P
Quick circle	Ctrl+Q
Redraw shape	Ctrl+R
Save shape file	Ctrl+S
Save screen	Ctrl+?
Tab setting	Tab
Tab move	Ctrl+Tab
Type at cursor	Ctrl+T

MAJOR VARIABLES

picture$	Name for screen picture.
movx	Cursor step for mode choice.
beam	Beam toggle.
beamtrip	Trips between B and Copy.
x and y	Horizontal and vertical cursor positions.
tes	Result of TEST pixel.
jump,jumpy	Fast horizontal, vertical cursor move.
xp,yp	Tab setting.
fil	Draw to Tab toggle.
tb	Tab set.
infil	Fill toggle.
movetab	Tab moving toggle.
extrax,extray	Tab move jump.
mem	Shape memory toggle.
memtrip	Start/stop memory.
radx,rady	Horizontal and vertical radii.
stpos	Circle start position.
side	Number of sides.
delay	Delay in polygon.
cw	Clockwise or anticlockwise.
startat	Start position, on circumference or centre.
tba	Draw to Tab.
poly	Circumference divided by number of sides.
rstep	Cursor moved in pixels (arrow option).
rtes	Radius option TEST.
sh,shape	Number of shape.
mag	Magnification or reduction.
space,minus	Positions for blank$(n).
character$(n)	Shape string.
blank$(n)	Position of blanks or minus in character$.

SUBROUTINES

50	Start, lowers memory and loads machine code.
350	Initialise, sets up the screen windows and colours.
500	Main program.
950	Quick circle.
1030	Pen colour change.
1220	Circles.
1950	Radius.
2050	Type at cursor.
2240	Fill.
2440	Ink change.
2640	Save screen to memory.
2760	Save memory to tape or disc.
2800	Cursor jump.
2920	Draw from memory.
3230	Save shapes.
3350	Input shapes.
3520	Load picture to memory.
3630	Go to option pages.
3690	Machine code, allows saving and recalling a picture to the reserved memory.

Listing I

```
50 REM ************************ EASY
DRAW ****************************
60 REM *** A screen drawing and paint
ing utility by Glynne Davies ***
70 REM
80 REM
90 REM
100 REM
110 MEMORY &665A : REM * reserve memo
ry for screen dump and disc drive *
120 GOSUB 3700: REM * load machine co
de for memory dump *
130 REM *** Title page ***
140 MODE 0:FOR n=-PI TO 3 STEP PI/16:
pt= (pt+1) MOD 7:pc=pt+1
150 TAG: PLOT 80+(20*SIN(n)),200+(120
*COS(n)),pc
160 PRINT "*--EASY DRAW--*";:NEXT:TAG
OFF
170 LOCATE 18,23:PRINT "BY":LOCATE 6,
25: PEN 3:PRINT "G.M.DAVIES"
180 FOR n=1 TO 6000:NEXT
190 CALL &BC02
200 MODE 2:INK 15,1:INK 0,13:INK 1,0:
x=286:y=150:beam =0:testab=3
210 jump=20:jumpy=20:DIM blank$(100):
DIM character$(40)
220 beam=0:LOCATE 17,12:INPUT "Do you
 want to load a picture to memory Y o
r N";pic$
230 IF pic$="Y" OR pic$="y" THEN LOCA
TE 20,14:INPUT "Type in the name of t
he picture";picture$:ELSE GOTO 250
240 LOAD picture$,&6675
250 CLS:LOCATE 20,12:INPUT "Please ty
pe in the mode required (ie. 0  or 1
)";modescreen:IF modescreen > 1 OR mo
descreen <0 THEN GOTO 250
260 IF modescreen =1 THEN movx= 2:ELS
E movx=4
270 IF modescreen >1 THEN GOTO 250
280 MODE modescreen:INK 1,24
290 LOCATE(4*(modescreen*2)+5),12: IN
PUT "PICTURE Y/N";Y$
300 CLS
310 IF Y$="Y" OR Y$="y" THEN CALL 262
15
320 GOSUB 360 : REM * initialize *
330 GOSUB 510 : REM * program *
340 END
350 REM *** Initialize - set up the s
creen ***
360 WINDOW #2,1,2,1,22:WINDOW #3,1,80
/movx,23,25:WINDOW #0,1,80/movx,23,25
:WINDOW #4,3,80/movx,1,22
370 PAPER #2,0:PAPER #3,15:CLS #2:CLS
 #3
380 ORIGIN 64,48,64,640,48,400
390 DRAW 574,1,3:DRAW 574,350,3:DRAW
1,350,3:DRAW 1,1,3
400 PEN #2,0:LOCATE #2,2,20:PRINT #2,
CHR$(143);
410 REM ** Draw colours **
420 FOR count=0 TO 15
430 PEN #2,count
440 LOCATE #2,1,count+1:PRINT #2,CHR$
(65+count);CHR$(233)
450 NEXT count
460 PEN #2,4:LOCATE #2,1,1:PRINT #2,"
A"
470 PEN #2,5:LOCATE #2,1,17:PRINT #2,
"+"
480 IF movx=2 THEN LOCATE #2,1,1:PRIN
T #2,"A":LOCATE #2,1,5:PRINT #2,"E":L
OCATE #2,1,9:PRINT #2,"I":LOCATE #2,1
,13:PRINT #2,"M"
490 shift$="SHIFT":FOR count=1 TO 5:L
OCATE #2,1,17+count:PRINT #2,MID$ (sh
ift$,count,1):NEXT
495 PEN #3,1:LOCATE #3,13,3:PRINT #3,
"Beam off";:REM beam off
500 REM *** program ***
505 ON ERROR GOTO 505
510 WHILE exit <1
520 IF INKEY (1)=0 THEN PLOT x,y,tes:
x=x+movx:IF movtab=1 THEN extrax=extr
ax+movx
530 IF INKEY (8)=0 THEN PLOT x,y,tes:
x=x-movx:IF movtab=1 THEN extrax=extr
ax-movx
540 IF INKEY (0)=0 THEN PLOT x,y,tes:
y=y+2:GOSUB 2250:IF movtab=1 THEN ext
ray=extray+2:REM gosub to fill routin
e
550 IF INKEY (2)=0 THEN PLOT x,y,tes:
y=y-2:GOSUB 2250:IF movtab=1 THEN ext
ray=extray-2
560 IF INKEY (1)=32 THEN PLOT x,y,tes
:x=x+jump:IF movtab=1 THEN extrax=ext
rax+jump
570 IF INKEY (8)=32 THEN PLOT x,y,tes
:x=x-jump:IF movtab=1 THEN extrax=ext
rax-jump
580 IF INKEY (0)=32 THEN PLOT x,y,tes
:y=y+jumpy:IF movtab=1 THEN extray=ex
tray+jumpy
590 IF INKEY (2)=32 THEN PLOT x,y,tes
:y=y-jumpy:IF movtab=1 THEN extray=ex
tray-jumpy
600 IF mem=1 AND xp=XPOS AND yp=YPOS
AND memtrip=1 THEN character$(sh)=cha
racter$(sh)+STR$(xrel)+STR$(yrel):mem
trip=0:IF LEN(character$(sh)) >240 TH
EN SOUND 1,300,25: PEN #3,1:LOCATE #3
,1,1:PRINT #3,"Last tab setting ":mem
=0
610 IF INKEY(50)=128 THEN SOUND 1,100
,20:GOSUB 2920:REM redraw memory shap
e
620 IF INKEY(36)=128 THEN SOUND 1,100
,15:GOSUB 3520: REM load picture into
 memory
630 IF INKEY(16)=32 THEN CLS #4:MOVE
0,0:DRAW 574,1,3:DRAW 574,350,3:DRAW
1,350,3:DRAW 1,1,3:MOVE x,y:REM * era
se picture *
640 IF INKEY (9)=128 THEN SOUND 1,120
,10:beamtrip=(beamtrip+1) MOD 2:IF be
amtrip=0 THEN PEN #3,1:LOCATE #3,13,3
:PRINT #3,"Beam off";:REM beam on/off
/on copy key
650 IF beamtrip=0 THEN GOTO 670
660 IF INKEY (9)=0 THEN beam=1:PEN #3
,4:LOCATE #3,13,3:PRINT #3,"Beam on "
;:ELSE beam=0:PEN #3,4:LOCATE #3,13,3
:PRINT #3,"Beam off";:REM beam on/off
/on copy key
670 IF beamtrip=1 THEN GOTO 690
680 IF INKEY (54)=128 THEN SOUND 1,10
0,10:beam=(beam+1) MOD 2:IF beam=1 TH
EN PEN #3,1:LOCATE #3,13,3:PRINT #3,"
Beam on ";:ELSE beam=0:PEN #3,1:LOCAT
E #3,13,3:PRINT #3,"Beam off";:REM be
am on/off
690 IF INKEY(68)=128 THEN SOUND 1,100
,20:movtab=(movtab+1) MOD 2:IF movtab
=1 THEN PEN #3,1:LOCATE #3,1,2:PRINT
#3,"Tab moving      ":ELSE PEN #3,1:
LOCATE #3,1,2:PRINT #3,"Tab stopped
   ":xp=xp+extrax:yp=yp+extray:extrax
=0:extray=0 :REM moving tab
700 IF INKEY(35)=128 THEN SOUND 1,100
,10:GOSUB 3350 : REM input from tape
710 IF INKEY (21)=32 THEN GOSUB 1040:
 REM * pen colours *
730 IF INKEY (45)=128 THEN SOUND 1,10
0,25:GOSUB 2800: REM cursor jump
740 IF INKEY (68)=0 THEN SOUND 1,100,
5:xrel=x-xp:yrel=y-yp:xp=x:yp=y:PEN #
3,1:LOCATE #3,1,2:PRINT #3,"X";XPOS;"
Y";YPOS:tb=1:memtrip=1:REM tab settin
g
750 IF sh> 39 THEN GOTO 770
760 IF INKEY (38)=128 THEN SOUND 1,10
0,5:mem=(mem+1) MOD 2:IF mem=1 THEN P
EN #3,1:LOCATE #3,1,1:PRINT #3,"Memor
y on shape";sh:ELSE PEN #3,1:LOCATE #
3,1,1:PRINT #3,"Memory is off      ";
sh=sh+1
770 IF INKEY (61)=128 THEN SOUND 1,10
0,5:fil=(fil+1) MOD 2:IF fil=1 THEN L
OCATE #3,1,1:PRINT #3,"Draw to tab on
```

```
  ":ELSE LOCATE #3,1,1:PRINT #3,"Draw
 to tab off ": REM draw to tab functi
on
780 IF INKEY(52)=128 THEN SOUND 1,200
,25:GOSUB 3630:REM go to options page
*
790 IF INKEY (68)=128 THEN SOUND 1,10
0,5: GOSUB 3230 : REM save shapes to
tape
800 IF INKEY (67)=128 THEN SOUND 1,10
0,25:PLOT x,y,14:GOSUB 960 :REM * Qui
ck circle *
810 IF INKEY (68)=32 THEN DRAWR xp+ex
trax-XPOS,yp+extray-YPOS,p :PLOT x,y,
14
820 IF INKEY (16)=128 THEN SOUND 1,10
0,10:PLOT x,y,14: GOSUB 1230 : REM *
circles, polygons *
830 IF INKEY (53)=128 THEN SOUND 1,10
0,5:infil=(infil+1) MOD 2:IF infil=1
THEN fill=TEST(x-2,y-2):LOCATE #3,1,1
:PRINT #3,"Fill on          ":ELSE LOC
ATE #3,1,1:PRINT #3,"Fill off
*
840 IF INKEY(27)=128 THEN SOUND 1,100
,5: GOSUB 2450 :REM * ink colour chan
ge *
850 IF INKEY (30)=128 THEN SOUND 1,10
0,5:MOVE 0,0:DRAW 574,1,3:DRAW 574,35
0,3:DRAW 1,350,3:DRAW 1,1,3:GOTO 2650
 : REM * picture save *
860 IF INKEY(51)=128 THEN SOUND 1,100
,5:GOSUB 2870 : REM * type at cursor
*
870 IF x=xx AND y=yy THEN GOTO 940
880 IF beam=1 THEN DRAW x,y,p:tes=p
890 IF beam=0 THEN tes=TEST(x,y):xx=x
:yy=y:PLOT x,y,14
900 IF tb=1 AND fil =1 THEN DRAWR  xp
+extrax-XPOS,yp+extray-YPOS,p
910 PEN #3,4:LOCATE #3,1,3:PRINT #3,"
X";x;"Y";y
920 IF xp=xpt AND yp=ypt AND extrax=x
extra AND extray=yextra THEN GOTO 940

930 PEN #3,1:LOCATE #3,1,2:PRINT #3,"
X";xp+extrax;"Y";yp+extray :xpt=xp:yp
t=yp:xextra=extrax:yextra=extray
940 WEND
950 REM ** Quick circle **
960 CALL &BB1B:IF INKEY(67)=128 THEN
GOTO 960:CALL &BB1B
970 qc=1:PEN #3,1:LOCATE #3,1,1:INPUT
 #3,"RADIUS(-) above 4 or cursor ente
r 0";radx$:IF radx$=" " OR radx$="" T
HEN GOTO 970
975 IF ASC(radx$) < 48 THEN GOTO 970
980 radx= VAL(radx$):LOCATE #3,1,1:PR
INT #3,SPACE$ (40):IF radx=0 THEN GOS
```

```
UB 1850
990 radx=radx+(radx MOD movx) :MOVE x
+radx,y
1000 FOR count=0 TO 360 STEP 2:DEG:DR
AW x+(radx*COS(count)),y+(radx*SIN(co
unt)),p:NEXT
1010 LOCATE #3,1,1:PRINT #3,SPACE$(40
)
1020 RAD:qc=0:RETURN
1030 REM ** pen colour change **
1040 IF INKEY (69)=32 THEN SOUND 1,15
0,5:p=0 :REM * A *
1050 IF INKEY (54)=32 THEN SOUND 1,15
0,5:p=1 :REM * B *
1060 IF INKEY (62)=32 THEN SOUND 1,15
0,5:p=2 :REM * C *
1070 IF INKEY (61)=32 THEN SOUND 1,15
0,5:p=3 :REM * D *
1080 IF INKEY (58)=32 THEN SOUND 1,15
0,5:p=4 :REM * E *
1090 IF INKEY (53)=32 THEN SOUND 1,15
0,5:p=5 :REM * F *
1100 IF INKEY (52)=32 THEN SOUND 1,15
0,5:p=6 :REM * G *
1110 IF INKEY (44)=32 THEN SOUND 1,15
0,5:p=7 :REM * H *
1120 IF INKEY (35)=32 THEN SOUND 1,15
0,5:p=8 :REM * I *
1130 IF INKEY (45)=32 THEN SOUND 1,15
0,5:p=9 :REM * J *
1140 IF INKEY (37)=32 THEN SOUND 1,15
0,5:p=10:REM * K *
1150 IF INKEY (36)=32 THEN SOUND 1,15
0,5:p=11:REM * L *
1160 IF INKEY (38)=32 THEN SOUND 1,15
0,5:p=12:REM * M *
1170 IF INKEY (46)=32 THEN SOUND 1,15
0,5:p=13:REM * N *
1180 IF INKEY (34)=32 THEN SOUND 1,15
0,5:p=14:REM * O *
1190 IF INKEY (27)=32 THEN SOUND 1,15
0,5:p=15:REM * P *
1200 PEN #2,p:LOCATE #2,2,20:PRINT #2
,CHR$(233);
1210 RETURN
1220 REM *** circles ***
1230 CALL &BB1B:IF INKEY(16)=128 THEN
 GOTO 1230:CALL &BB1B
1240 PEN #3,10:LOCATE #3,1,1:INPUT #3
,"RADIUS(-) above 4 or cursor enter 0
";radx$:IF radx$= " " OR radx$="" THE
N GOTO 1240
1245 IF ASC(radx$) < 48 THEN GOTO 124
0
1250 radx= VAL(radx$)
1260 LOCATE #3,1,1:PRINT #3,SPACE$ (4
0):IF VAL(radx$) < 4 THEN GOSUB 1850:
GOTO 1300
1270 LOCATE #3,1,1:PRINT #3,SPACE$ (4
0)
1280 LOCATE #3,1,1:INPUT #3,"RADIUS u
p/down >2";rady$:IF rady$=" " OR rady
$="" THEN GOTO 1280
1285 IF ASC(rady$) < 48 THEN GOTO 128
0
1290 rady=VAL(rady$):IF rady <2 THEN
GOSUB 1940
1300 LOCATE #3,1,1:PRINT #3,SPACE$ (6
0)
1310 LOCATE #3,1,1:INPUT #3,"Start po
sition 0 to 1 (ie .65)";stpos$:IF stp
os$=" " OR stpos$="" THEN GOTO 1310
1315 IF ASC(stpos$) < 46 THEN GOTO 13
10
1320 stpos=VAL(stpos$):IF VAL(stpos$)
 >1 THEN LOCATE #3,1,1:PRINT #3,"BETW
EEN 0 and 1 ie. .1 .25 .7 .9":FOR n=1
 TO 1000:NEXT:GOTO 1310
1330 LOCATE #3,1,1:INPUT #3,"Number o
f sides. Go >70 for circles";side$:IF
 side$=" " OR side$="" THEN GOTO 1330
1335 IF ASC(side$) <48 THEN GOTO 1330

1340 side=VAL(side$):IF VAL(side$) <
3 THEN GOTO 1330
1350 poly=(2*PI)/side
```

```
1360 LOCATE #3,1,1:PRINT #3,SPACE$ (4
0)
1365 IF side > 19 THEN GOTO 1380 :REM
 miss delay
1370 IF side < 20 THEN LOCATE #3,1,1:
INPUT #3,"state delay ie. 200 for par
t shape";delay$:IF delay$=" " OR dela
y$="" THEN GOTO 1370
1375 IF side < 20 AND ASC(delay$) < 4
8 THEN GOTO 1370: ELSE delay=VAL(dela
y$):LOCATE #3,1,1:PRINT #3, SPACE$(40
)
1380 LOCATE #3,1,1:INPUT #3,"Clockwis
e plotting 0 Anticlockwise 1";cw$:IF
cw$=" " OR cw$="" THEN GOTO 1380
1385 IF ASC(cw$) <48 THEN 1380
1390 cw=VAL(cw$):IF VAL(cw$) >1 THEN
GOTO 1380
1400 LOCATE #3,1,1:PRINT #3,SPACE$ (4
0)
1410 LOCATE #3,1,1:INPUT #3,"Plot fro
m the cursor 1 or the centre 0";start
at$:IF startat$=" " OR startat$="" TH
EN GOTO 1410
1415 IF ASC(startat$) < 48 THEN GOTO
1410
1420 startat=VAL(startat$):IF VAL(sta
rtat$) >1 THEN GOTO 1410
1430 IF tb=0 THEN GOTO 1460
1440 LOCATE #3,1,1:INPUT #3,"Draw to
tab setting 1 or 0 for off (3D)";tba$
:IF tba$=" " OR tba$="" THEN GOTO 144
0
1445 IF ASC(tba$) <48 THEN GOTO 1440
1450 tba=VAL(tba$):IF VAL(tba$) > 1 T
HEN GOTO 1440
1460 LOCATE #3,1,1:PRINT #3,SPACE$ (4
0)
1470 LOCATE #3,1,1:PRINT #3,"SPACE BA
R TO STOP"
1475 radx=radx+(radx MOD movx)
1480 IF cw=1 THEN GOTO 1680: REM anti
clockwise
1490 IF startat=0 THEN PLOT x,y,tes
1500 IF startat=1 THEN x=x-(radx* SIN
(stpos*2*PI)):y=y-(rady*COS(stpos*2*P
I)):MOVE x,y
1510 MOVE x+(radx*SIN(stpos*2*PI)),y+
(rady*COS(stpos*2*PI))
1520 REM ** circle clockwise **
1530 FOR count= stpos*2*PI TO (4*PI)+
poly STEP poly
1540 DRAW x+(radx*SIN(count)),y+(rady
*COS(count)),p:IF movtab=1 THEN extra
x=(XPOS-x)-radx*SIN(stpos*2*PI):extra
y=(YPOS-y)-rady*COS(stpos*2*PI)
1550 IF INKEY (68)=32 THEN DRAWR xp+e
xtrax-XPOS,yp+extray-YPOS,p:MOVE x+(r
adx*SIN(count)),y+(rady*COS(count))
1560 IF INKEY (21)=32 THEN GOSUB 1840
: REM * pen colours *
1570 IF tba=1 THEN DRAWR xp+extrax-XP
OS,yp+extray-YPOS,p:MOVE x+(radx*SIN(
count)),y+(rady*COS(count))
1580 IF INKEY (47)=0 THEN count=(4*PI
)+poly
1590 IF xyoff=1 THEN GOTO 1610
1600 LOCATE #3,1,3:PRINT #3,"X";XPOS;
"Y";YPOS
1610 FOR rest=0 TO delay:IF INKEY (47
)=0 THEN count=(4*PI)+poly:ELSE NEXT
1620 NEXT
1630 LOCATE #3,1,1:PRINT #3,SPACE$(60
):x=XPOS:y=YPOS
1640 LOCATE #3,1,3:PEN #3,4:PRINT #3,
"X";XPOS;"Y";YPOS:IF beam=1 THEN PEN
#3,1:LOCATE #3,13,3:PRINT #3,"Beam on
 ";:ELSE beam=0:PEN #3,1:LOCATE #3,13
,3:PRINT #3,"Beam off";:REM beam on/o
ff
1650 extrax=CINT(extrax):extray=CINT(
extray)
1660 RETURN
1670 REM ** circle anticlockwise **
1680 stpos=stpos+0.5:IF startat=0 THE
N PLOT x,y,tes
1690 IF startat=1 THEN x=x+(radx* SIN
(stpos*2*PI)):y=y+(rady*COS(stpos*2*P
I))
1700 MOVE x-(radx*SIN(stpos*2*PI)),y-
(rady*COS(stpos*2*PI))
1710 FOR count=2*PI*stpos TO -((2*PI)
+poly) STEP -poly
1720 DRAW x-(radx*SIN(count)),y-(rady
*COS(count)),p :IF movtab=1 THEN extr
ax=(XPOS-x)+radx*SIN(2*PI*stpos):extr
ay=(YPOS-y)+rady*COS(2*PI*stpos)
1730 IF INKEY (68)=32 THEN DRAWR xp+e
xtrax-XPOS,yp+extray-YPOS,p:MOVE x-(r
adx*SIN(count)),y-(rady*COS(count))
1740 IF tba=1 THEN DRAWR xp+extrax-XP
OS,yp+extray-YPOS,p:MOVE x-(radx*SIN(
count)),y-(rady*COS(count))
1750 IF INKEY (47)=0 THEN count= -((2
*PI)+poly)
1760 LOCATE #3,1,3:PRINT #3,"X";XPOS;
"Y";YPOS
1770 IF INKEY (21)=32 THEN GOSUB 1840
: REM * pen colours *
1780 FOR rest=0 TO delay:IF INKEY (47
)=0 THEN count= -((2*PI)+poly):ELSE N
EXT
1790 NEXT
1800 LOCATE #3,1,1:PRINT #3,SPACE$(60
):x=XPOS:y=YPOS:count =0
1810 LOCATE #3,1,3:PEN #3,4:PRINT #3,
"X";XPOS;"Y";YPOS:IF beam=1 THEN PEN
#3,1:LOCATE #3,13,3:PRINT #3,"Beam on
 ";:ELSE beam=0:PEN #3,1:LOCATE #3,13
,3:PRINT #3,"Beam off";:REM beam on/o
ff
1820 extrax=CINT(extrax):extray=CINT(
extray)
1830 RETURN
1840 REM * horizontal cursor radius s
etting *
1850 WHILE radx =0
1860 LOCATE #3,1,1:PRINT #3,"Use arro
w keys <--> to place,then enter";
1870 IF INKEY(1)=0 THEN PLOT x+rstep,
y,rtes:rtes=TEST(x+rstep+movx,y):rste
p=rstep+movx:PLOT x+rstep,y,14
1880 IF INKEY (8)=0 THEN PLOT x+rstep
,y,rtes:rtes=TEST(x+rstep-movx,y):rst
ep=rstep-movx:PLOT x+rstep,y,14
1890 IF INKEY (18)=0 THEN radx=ABS(rs
tep):PLOT x+rstep,y,rtes
1900 PLOT x,y,14
1910 WEND:rtes=14:rstep=0:CALL &BB18
1920 IF qc=1 THEN RETURN
1930 LOCATE #3,1,3:PRINT #3,SPACE$(20
):LOCATE #3,1,3:PRINT #3,"Hori-rad is
";radx;"y=X"
1940 rady=0
1950 REM * up/down radius set by curs
or *
1960 WHILE rady=0
1970 LOCATE #3,1,1:PRINT #3,"Use arro
w keys then press enter "
1980 IF INKEY(0)=0 THEN PLOT x,y+rste
p,rtes:rtes=TEST(x,y+rstep+2):rstep=r
step+2:PLOT x,y+rstep,14
1990 IF INKEY(2)=0 THEN PLOT x,y+rste
p,rtes:rtes=TEST(x,y+rstep-2):rstep=r
step-2:PLOT x,y+rstep,14
2000 IF rstep=0 AND INKEY(63)=0 THEN
rady=radx
2010 IF INKEY(18)=0 THEN rady=ABS(rst
ep):PLOT x,y+rstep,rtes
2020 PLOT x,y,14
2030 WEND:rstep=0:rtes=0:CALL &BB00
2040 CALL &BB18:RETURN
2050 REM *** type at cursor ***
2060 CALL &BB1B
2070 LOCATE #3,1,1:PRINT #3,"   Type
at cursor"
2080 LOCATE #3,1,1:PRINT #3,"   Type
at cursor"
2090 LOCATE #3,1,2:PRINT #3,"   (Ente
r) to end"
2100 FOR n=1 TO 500:NEXT:CALL &BB00
2110 WHILE t<1
2120 type$=INKEY$
2130 IF INKEY(18)=0 THEN PLOT x,y,0:
t=1:papr=0:GOTO 2200
2140 TAG:PLOT x,y,p
2150 IF type$="" GOTO 2190
```

```
2160 IF ASC(type$)=127 THEN PLOT x,y,
0: x=x-(movx*4):MOVE x,y:PRINT " ";:G
OTO 2190
2170 PRINT type$;
2180 x=x+(movx*8):MOVE x,y
2190 type$=""
2200 WEND
2210 t=0:LOCATE #3,1,1:PRINT #3,SPACE
$(40)
2220 TAGOFF
2230 RETURN
2240 REM ** fill routine **
2250 IF infil = 0 THEN RETURN
2260 WHILE lin < 1
2270 IF TEST (x,y) <> filc THEN lin =
1:GOTO 2330
2280 incl=incl+movx:telf=TEST (x-incl
,y)
2290 IF telf <> filc THEN MOVE x,y:DR
AW x-(incl-movx),y,p:lin =1
2300 IF x<2 THEN lin=1
2310 IF y<2 THEN lin =1
2320 IF y>348 THEN lin=1
2330 WEND
2340 lin=0
2350 WHILE lin < 1
2360 incr=incr+movx:terf=TEST (x+incr
,y)
2370 IF x>575 THEN lin=1
2380 IF y<2 THEN lin=1
2390 IF y>348 THEN lin=1
2400 IF terf <> filc THEN MOVE x,y:DR
AW x+(incr-movx),y,p:lin =1
2410 WEND
2420 lin=0:incl=0:incr=0:telf=17:terf
=17
2430 RETURN
2440 REM ** ink change **
2450 CALL &BB1B:IF INKEY(27)=128 THEN
 2450:CALL &BB1B
2455 LOCATE #3,1,1:INPUT #3,"Ink chan
ge Y/N";y$:IF y$="Y" OR y$="y" THEN
GOTO 2460:ELSE LOCATE #3,1,1:PRINT #3
,SPACE$(40):RETURN
2460 LOCATE #3,1,1:INPUT #3,"   Pen t
o change";in$:IF in$="" THEN GOTO 24
60 2470 IF ASC(in$) < 65 OR ASC(in$)
>112 THEN LOCATE #3,1,1:PRINT #3,SPAC
E$(40):GOTO 2450
2470 IF ASC(in$) < 65 OR ASC(in$) >11
2 THEN LOCATE #3,1,1:PRINT #3,SPACE$(
40):GOTO 2450
2480 in$=UPPER$(in$):IF ASC(in$) > 80
 THEN GOTO 2450
2490 p=ASC (in$)-65
2500 IF p>15 THEN GOTO 2450
2510 CLS #3
2520 WHILE inchange <1
2530 LOCATE #3,1,1: PRINT #3,"Ink cha
nge (Shift)"
2540 PEN #3,p
2550 LOCATE #3,4,2: PRINT #3,"To set
(Enter)"
2560 LOCATE #3,1,3: PRINT #3,"     IN
K";p;",";i;
2570 IF INKEY(21)=32 THEN i=(i+1) MOD
 27
2580 IF INKEY(18)=0 OR INKEY (6)=0 TH
EN inchange=1
2590 INK p,i
2600 WEND
2610 i=0:inchange=0
2620 CLS #3:PEN #3,4:LOCATE #3,1,3:PR
INT #3,"X";XPOS;"Y";YPOS
2630 RETURN
2640 REM ** save screen to memory **
2650 FOR n=1 TO 100:NEXT:CALL &BB00
2660 PAPER #2,15:CLS #2:CLS #3
2670 CALL 26203
2680 CLS #4
2690 FOR n=1 TO 2000:NEXT n
2700 CALL 26215
2710 FOR n=1 TO 2000:NEXT
2720 CALL &BB00
2730 LOCATE 2,23:INPUT"Save screen Y/
N";ipt$
2740 IF ipt$="Y" OR ipt$="y" THEN GOS
UB 2770
2750 MODE 1:y=y+2:LOCATE 12,12:INPUT
"End EASYDRAW Y/N";y$:IF y$="y" OR y$
="Y" THEN CALL &BC02:MODE 1:LOCATE 6,
2:PRINT "Easydraw program has finishe
d":END
2755 MODE 2:GOTO 220
2760 REM ** Save to tape **
2770 SPEED WRITE 1:LOCATE 1,23:INPUT
"Please print name  ";name$
2780 SAVE name$,B,&6675,&4000
2790 RETURN
2800 REM *** cursor jump distance ***
2810 CALL &BB1B:IF INKEY(45)=128 THEN
 GOTO 2810:CALL &BB1B
2820 CALL &BB00
2830 PEN #3,1:LOCATE #3,1,1:INPUT #3,
" Type in horizontal cursor jump";jum
p$:IF jump$=" " OR jump$="" THEN GOTO
 2830
2835 IF ASC(jump$) <48 THEN GOTO 2830
2840 IF VAL(jump$)<4 THEN LOCATE #3,1
,1:PRINT #3,SPACE$(40):GOTO 2830
2850 jump=VAL(jump$)
2860 LOCATE #3,1,1:INPUT #3,"Type in
the vertical cursor jump ";jumpy$:IF
jumpy$=" " OR jumpy$="" THEN 2860
2865 IF ASC(jumpy$) <48 THEN GOTO 286
0
2870 IF VAL(jumpy$)<2 THEN LOCATE #3,
1,1:PRINT #3,SPACE$(40):GOTO 2860
2880 jumpy= VAL(jumpy$)
2890 LOCATE #3,1,1:PRINT #3,SPACE$(40
)
2900 LOCATE #3,1,3:PRINT #3,SPACE$(20
):LOCATE #3,1,3:PEN #3,4:PRINT #3,"X"
;XPOS;"Y";YPOS:IF beam=1 THEN PEN #3,
1:LOCATE #3,13,3:PRINT #3,"Beam on ";
:ELSE beam=0:PEN #3,1:LOCATE #3,13,3:
PRINT #3,"Beam off";:REM beam on/off
2910 RETURN
2920 REM draw from memory
2930 CALL &BB1B:IF INKEY(50)=128 THEN
 2930:CALL &BB1B
2940 LOCATE #3,1,1:PRINT #3,SPACE$(40
)
2950 IF sh=0 THEN LOCATE #3,1,1:PRINT
 #3,"No shapes in memory":FOR n=1 TO
1000:NEXT:LOCATE #3,1,1:PRINT #3, SPA
CE$(40):RETURN
2960 LOCATE #3,1,1:INPUT #3,"Type sha
pe number";shape$:IF shape$=" " OR sh
ape$="" THEN GOTO 2960
2965 IF ASC(shape$) <48 THEN GOTO 296
0
2970 LOCATE #3,1,1:PRINT #3,SPACE$(40
)
2980 IF VAL(shape$) >= sh THEN LOCATE
 #3,1,1:PRINT #3,"Not available 0 to"
;sh-1:FOR n=1 TO 1000:NEXT:GOTO 2940
```

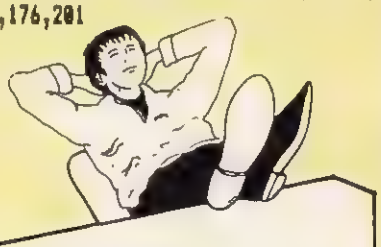
```
2990 shape= VAL(shape$)
3000 LOCATE #3,1,1:PRINT #3,SPACE$(40
)
3010 LOCATE #3,1,1:INPUT #3,"At magni
fication";mag$:mag= VAL(mag$)
3020 LOCATE #3,1,1:PRINT #3,SPACE$(40
)
3030 IF mag < 0.1 THEN LOCATE #3,1,1:
PRINT #3,"A little to small  ":FOR n=
 1 TO 1000:NEXT:GOTO 3010
3040 c=0
3050 LOCATE #3,1,1:PRINT #3,SPACE$(40
)
3060 IF INSTR(character$(shape)," - "
)> 1 THEN GOTO 3080
3070 character$(shape)=character$(sha
pe)+" - "
3080 count=0
3090 WHILE count< (LEN(character$(sha
pe))-3)
3100 count=count+1
3110 space=INSTR(count,character$(sha
pe)," "):minus=INSTR(count,character$
(shape),"-")
3120 IF space < minus THEN count=spac
e
3130 IF space > minus THEN count=minu
s
3140 blank$(c)=STR$(count):c=c+1
3150 WEND
3160 REM draw from memory
3170 FOR count=2 TO c-3 STEP 2
3180 drax= VAL(MID$(character$(shape)
,VAL(blank$(count)),VAL(blank$(count+
1))-VAL(blank$(count))))
3190 dray= VAL(MID$(character$(shape)
,VAL(blank$(count+1)),VAL(blank$(coun
t+2))-VAL(blank$(count+1))))
3200 DRAWR mag*drax,mag*dray,p
3210 NEXT count
3220 count=0:RETURN
3230 REM save shapes to tape
3240 CALL &BB1B: IF INKEY(60)=128 THE
N GOTO 3240:CALL &BB1B
3250 LOCATE #3,1,1:INPUT #3, "Save sh
apes Y/N";satap$:LOCATE #3,1,1:PRINT
#3,SPACE$(40):IF satap$="Y" OR satap$
="y" THEN GOTO 3260:ELSE GOTO 3340
3260 LOCATE #3,1,1:INPUT #3, "File na
me";file$
3270 IF LEN(file$) > 8 THEN LOCATE #3
,1,1:PRINT #3,"Below eight letters":F
OR n=1 TO 300:LOCATE #3,1,1:PRINT #3,
SPACE$(40):GOTO 3260
3280 OPENOUT file$
3285 PRINT #9,sh
3290 FOR count=0 TO sh
3300 PRINT #9,character$(count)
3310 NEXT count
3320 CLOSEOUT
3330 CLS #3:PEN #3,4:LOCATE #3,1,3:PR
INT #3,"X";XPOS;"Y";YPOS;
3340 RETURN
3350 REM input file for shapes
3360 CALL &BB1B:IF INKEY(35)=128 THEN
 GOTO 3370:CALL &BB1B
3370 LOCATE #3,1,1:INPUT #3, "Load sh
apes Y/N";lotap$:LOCATE #3,1,1:PRINT
#3,SPACE$(40):IF lotap$="Y" OR lotap$
="y" THEN GOTO 3380:ELSE GOTO 3510
3380 LOCATE #3,1,1:INPUT #3, "File na
me";file$
3390 IF LEN(file$) > 8 THEN LOCATE #3
,1,1:PRINT #3,"Below eight letters":F
OR n=1 TO 300:LOCATE #3,1,1:PRINT #3,
SPACE$(40):GOTO 3380
3400 OPENIN file$
3405 INPUT #9,sh
3410 FOR count =0 TO sh
3420 INPUT #9,character$(count)
3430 NEXT count
3470 FOR count=0 TO sh
3480 IF LEFT$(character$(count),1)="-
" THEN GOTO 3490:ELSE character$(coun
t)=" "+character$(count)
3490 NEXT count:count=0
3500 CLS #3:PEN #3,4:LOCATE #3,1,3:PR
INT #3,"X";XPOS;"Y";YPOS;
3510 RETURN
3520 REM load picture to memory
3530 CALL &BB1B:IF INKEY(36)=128 THEN
 GOTO 3530: CALL &BB1B
3540 LOCATE #3,1,1:PRINT #3,SPACE$(40
)
3550 LOCATE #3,1,1:INPUT #3,"Load in
picture  Y/N ";pict$
3560 LOCATE #3,1,1:PRINT #3,SPACE$(40
)
3570 IF pict$="Y" OR pict$="y" THEN G
OTO 3580:ELSE RETURN
3580 LOCATE #3,1,1:INPUT #3,"Print pi
cture name ";picture$
3590 LOCATE #3,1,1:PRINT #3,SPACE$(40
)
3600 LOAD picture$,&6675
3610 CLS #3
3620 RETURN
3630 REM go to option pages
3640 CALL &BB1B:IF INKEY(52)=128 THEN
 GOTO 3640:CALL &BB1B
3650 LOCATE #3,1,1:PRINT #3,SPACE$(40
)
3660 LOCATE #3,1,1:INPUT #3,"Go to op
tions Y/N";yes$:IF yes$="Y" OR yes$="
y" THEN MODE 2:y=y+2: GOTO 220: REM g
o to option pages
3670 LOCATE #3,1,1:PRINT #3,SPACE$(40
)
3680 RETURN
3690 REM ** save picture to memory ma
chine code **
3700 FOR n=26203 TO 26226
3710 READ x
3720 POKE n,x
3730 NEXT n
3740 RETURN
3750 DATA 1,0,64,33,0,192,17,117,102,
237,176,201
3760 DATA 1,0,64,33,117,102,17,0,192,
237,176,201
```

Listing II

```
10 MODE 0:INK 0,13:INK 15,1
20 LOAD "!filename",&C000
```

Listing III

```
10  MEMORY &665A :REM save memory for
 picture and disc drive
20  GOSUB 3700: REM save picture to m
emory machine code
30 LOAD "filename",&6675 :REM load pi
cture into memory
40 MODE 0:INK 14,9:INK 0,13:INK 15,1:
REM set up screen mode and colours
50 CALL 26215 : REM picture to screen
60 FOR n= 1 TO 2000:NEXT:REM wait
70 DRAW 640,400,3 :FOR n= 1 TO 1000:N
EXT: REM draw line across picture
80 CALL 26203 : REM save new picture
90 CLS:FOR n= 1 TO 1000:NEXT
100 CALL 26215 :REM new updated pictu
re
110 END
3690 REM ** save picture to memory ma
chine code **
3700 FOR n=26203 TO 26226
3710 READ x
3720 POKE n,x
3730 NEXT n
3740 RETURN
3750 DATA 1,0,64,33,0,192,17,117,102,
237,176,201
3760 DATA 1,0,64,33,117,102,17,0,192,
237,176,201
```

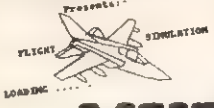


Give your fingers a rest . . .

All the listings from this month's issue are available on cassette.

See our special offer on Page 77.

MYRDDIN

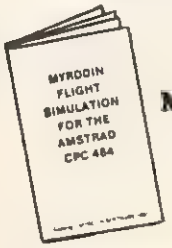


3D LANDMARKS YOU CAN FLY AROUND

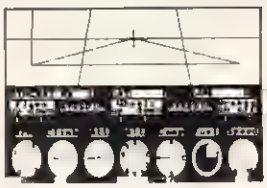
SUPERB REAL TIME SIMULATION

MYRDDIN FLIGHT SIMULATION

AMSTRAD CPC 464

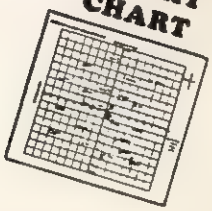


MANUAL

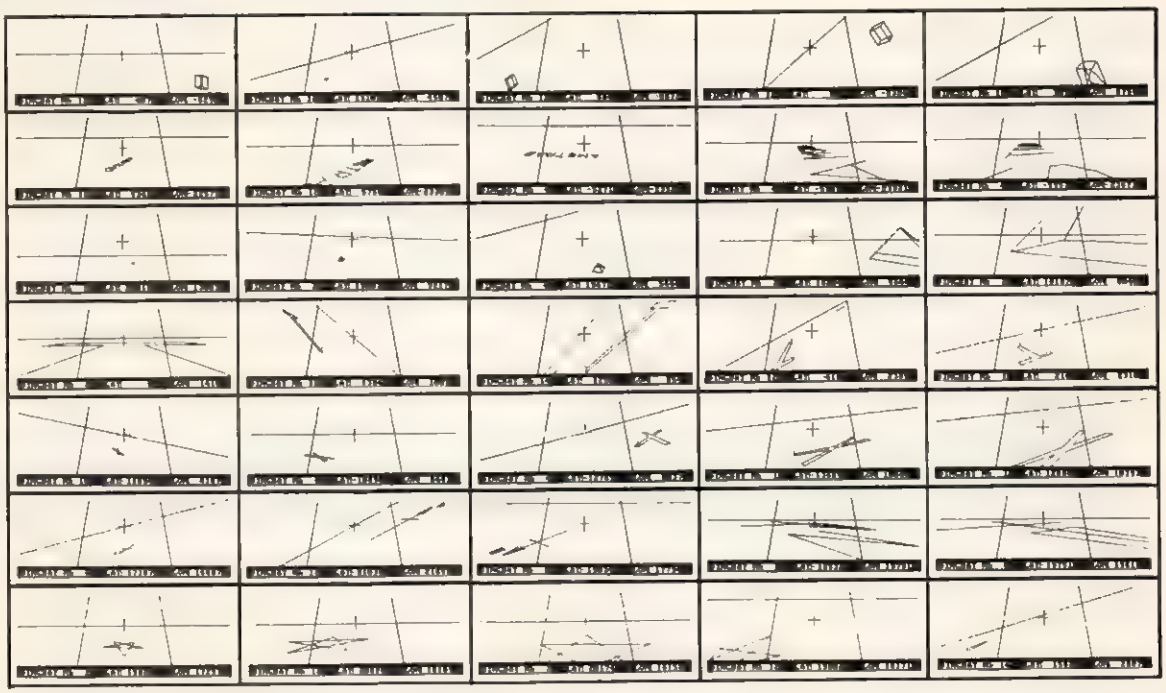


FULL SCREEN DISPLAY

FLIGHT CHART



Here are some screens from a typical flight showing the view from the cockpit (top half of screen) produced as printouts of the actual simulator.



A real time simulation with 3D graphics uses a massive 64000 x 64000 longitude & latitude flying area, making each flight completely different. Developed under pilot instruction to give realistic flight effect. The view through the cockpit gives moving 3D graphics.

Comprehensive instrument panel with moving needle meters & digital displays. 15 aircraft types with varying control sensitivities & speeds of between 100 – 500 knots.

3 runways available for refuelling, take off & landing. Ground and landmark orientation correct with all flying attitudes (rolls etc.).

The 3D graphics are still accurate when you fly upside down.

3D landmarks you can fly around.

Comes complete with manual & fully detailed chart of landmarks & airfields.

Joystick or keyboard operation.
Cassette version £11.95
Disc version £15.95
If your local dealer doesn't have it in stock yet, order from us direct.
For despatch within 48 hrs.
(usually 24 hrs.).

MYRDDIN SOFTWARE, PO BOX 61, SWINDON, WILTS.
Telephone: (0793) 40661

Please send me ..... Flight Simulator(s) by return of post for the Amstrad CPC 464

Name .....

Address .....

..... Postcode .....

Cheque enclosed for cassette £11.95/£15.95 disc (inc. P.P.)
OR Debit my Access A/C No:-

☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐

OR Telephone through your Access Order.

Signed.....

Aleatoire explains the thinking behind a classic computer challenge .

NIM—SUCH A

ONE of the oldest computer games is Nim, which was demonstrated at the 1951 Festival of Britain on a machine called Nimrod.

Journalists reported that the machine was so popular – particularly when it beat the German Ambassador five times in a row – that a nearby exhibition providing free drinks was practically deserted.

The game is very simple. Take a box of matches and lay the contents out in a random number of piles, each containing a random number of matches.

Two players alternate, and each one can take as many matches as he likes from any single pile. The object of the game is to make your opponent pick up the last match.

This game was analysed by a Charles Bouton of Harvard University about 1900 and he showed that there is a simple strategy which plays the game perfectly.

Every possible position is either "safe" or "unsafe". Any move from a safe position always creates an unsafe position, and there is *always* at least one move from an unsafe position which creates a safe position.

The proof of this is quite complex, but all you, or a computer, need to know is how to create safe positions. This is done by "matching the binarys" – that is, by making sure that there are an even number of groups of powers of 2 – an even number of 1s, 2s, 4s, 8s, 16s etc.

A simple example to clarify this rule. Suppose you have four piles containing 3, 5, 7 and 11 matches, then, in binary:

	8	4	2	1
3 =			1	1
5 =		1		1
7 =		1	1	1
11 =	1		1	1

Here we see that there is an even number of 1s, an even number of 4s but an odd number of 2s and 8s. The

Thoughts on number

MICRO-Maths by Keith Devlin *(Macmillan)* is subtitled "Mathematical problems and theorems to consider and solve on a computer".

The author writes a regular fortnightly column for The Guardian and includes "whatever I find fun and of interest", which appears to be mainly number theory, the Queen of Mathematics, of which the definitive book is by Beiler.

Unfortunately Devlin appears to have little interest in demonstrating, with programs, the problems he considers – the "and solve on a computer" of the title is left entirely to the reader.

Indeed, the only techniques he mentions are multilength working and fast multipliers – almost everything is treated from a maths point of view and various errors and misconceptions are included.

For example (on page 3) "the fastest computers currently – can perform something like 200 billion arithmetic operations per second". Devlin does not define "billion", so we British must assume he means that machines can add (using the simplest arithmetic operation) 200,000,000,000,000 numbers per second! He probably meant "million" but the correct term is megaflop or gigaflop.

Another niggle is on page 18: "Modern computer speeds are such that 50 or so users can be accessing the machine at the same time without being aware that they are not alone".

This is not true. Most multi-user computers begin to overload and degrade at the 50 user level until, at 70 users, they spend almost all of the time swopping between the users and no useful work is done. This phenomenon is called "saturation", is quite dramatic and the users are VERY aware of it.

A "heavily used" computer is usually run just below this sudden degradation point and certainly does not "spend most of the time sitting idle (and) waiting".

We are then told to "Remember, today's computers are capable of performing millions of instructions per second", that is megaflops. As he corrects a previous error he introduces a few more – a common computing experience. The above errors are part of an article on finding formulae to generate prime numbers.

In America "heavily used" computers apparently do have 10 idle hours a day (lucky them) so programs to soak up this time are written, usually to prove that the system is overloaded and that more equipment must be bought.

These are often very simple programs which search for extremely large, useless numbers to get the programmer/machine into the Guinness Book of Records. The results are often a triumph of computation over common sense and disowned by many pure mathematicians.

Nevertheless I envy the two 15-year-old Americans who got 350 hours on a Cyber 174 to find a record prime in 1978. I understand the program but still don't know how they managed to get the time. Such performances rely, not on the individual human mind, but on the incredible but often misunderstood power of the modern computer.

The book deals with a number of famous, though simple, computations of gigantic numbers such as PI, primes and perfect numbers,

SIMPLE GAME

position can therefore be made safe by removing 10 matches from the pile of 11 matches, and is the *only* correct move.

If you play this rule *all* the time then it guarantees that *you* will pick up the last match. To play the reverse game, that is to make your opponent pick up the last match, then play the rule *until* only one pile has more than one match – this is inevitable – and then make the move that produces an odd number of one-match piles.

If you are faced with a safe position then take just one match from the largest pile and hope your opponent will make an error.

A randomiser can be incorporated into the algorithm because sometimes there is more than one way of creating a safe position.

The above method is the way that many computers have been programmed to play this game. Once done a computer is quite capable of playing a game with a thousand piles of over a million matches each and, if a time limit were set, it would easily beat all humans whether they know the method or not.

To give you some idea of safe and unsafe positions, consider an even simpler version of the game where there is a single pile of N matches and players alternate in taking any number of matches up to half the pile, that is <=N/2. Again the player who picks up the last match is the loser.

Program 1 plays this game perfectly. Note that because it is an expert it gives an opponent the first move 80 per cent of the time, which means that you should eventually beat it fairly almost 80 per cent of the time.

To win all the time take less than one match, which effectively adds them to the pile but the expert doesn't notice unless you add line 55:

```
55 IF M<1 THEN 40
```

Finally, try modifying the program so that whoever gets the last match is now the winner.

```
10 N=INT(RND*1000)+100
20 PRINT "There are "N" matches at th
e start"
30 IF RND>0.8 THEN 80
40 INPUT "Your move ";M
50 IF M>N/2 THEN 40
60 N=N-M
70 PRINT "You have left "N
80 I=1
90 I=I*2
100 IF I<=N THEN 90
110 I=I/2-1
120 IF 2*I<N THEN I=N-1
130 PRINT "I take "N-I" leaving "I
140 N=I
150 IF N>1 THEN 40
160 IF N=1 THEN PRINT "I win" ELSE PR
INT "I lose"
170 PRINT"Another game": GOTO 10
```

Program 1

crunchers

Fermat's Last Theorem, the four-colour map problem, irrational numbers (though no mention of the Golden Ratio) Archimedes' Cattle Problem.

All these computations are so huge that it is a grain of wheat to 2 ^ 64 that you, with your puny micro and interpretive Basic, have no chance of repeating, let alone surpassing, the results. The book is therefore interesting but not very practical.

Take the cattle problem, for example. Devlin appears rather ignorant of this famous Pellian equation – that is equations of the form:

X*X – D*Y*Y = 1

Invented by Fermat, the problem is, given D, to find the smallest integer values of X and Y. If D=92 then X=1151 and Y=120 are the smallest.

For the cattle problem D=410,286,423,278,424 and Y is a 206,531 digit integer!

An American engineer, A. Bell, and two friends spent four years calculating the first 32 digits in 1889 but the full result was not obtained until 1965 on an IBM 7040. This answer covered 42 pages of printout and one wry comment was that the computers of the French Compagnie Bull would have a more appropriate choice. If you want to try a tricky Pellian set D=61.

The answer is X=1,766,319,... and calculating Y is left as an exercise for the reader to consider.

One other fascination of computational maths is the personalities who first invented or attempted the problems. Apart from Fermat, all of them seem to have monumental egos.

A story is told about Hardy, an English mathematician, sending a postcard claiming to have solved a famous conjecture before making a dangerous journey. He argued that God would not let him perish because he would then attain undeserved immortality.

Surely God could have destroyed the postcard just as easily, but Hardy would probably have replied that maybe even God didn't know the answer. A few years later Godel showed that certain problems, although they must have an answer, are nevertheless undecidable.

Give your fingers a rest . . .

All the listings from this month's issue are available on cassette.

See our special offer on Page 77.

Game of the Month

MR HUMPY has the hump. The beautiful Esmerelda has been captured by the French when they invaded Lincolnshire, and she has been taken to the dark and damp Boston-de-Stump.

You must guide Mr Humpy over the roof of the Stump and rescue Esmerelda from her cell.

However this is no easy matter as you will have to avoid lumps of rock, jump over chasms using only a slippery rope, cross moving drawbridges and fight off Frenchmen.

Also on each screen is a barrel of TNT which is about to explode. If it does you lose a life.

To move onto the next screen you must ring the bell. When you ring the bell on the 15th screen you get a super bonus and Mr Humpy is returned to Screen 1 – where the fuse on the TNT is burning faster.

KEYS

Z – Right
X – Left
Space – Jump

By ARAMELLO CHAPMAN

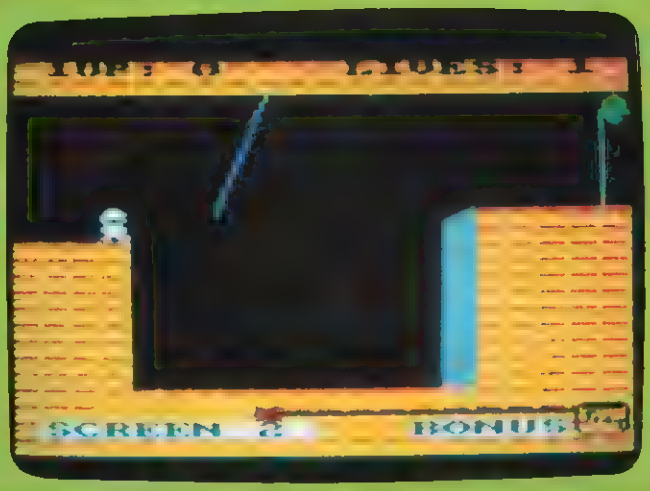
ROUTINES

140	Main loop. Go to the various hazard moving routines. Mr Humpy moving routines.
230	Bonus count, decreases bonus by (1/bonus) and prints fuse.
300	Makes TNT explode.
350	Moves Humpy. Reads keyboard. If jump key pressed to jump routines.
560	Makes Humpy jump, tests to see if he is on rope screen.
800	Updates position of fire balls.
970	Updates position of guard poles.
1070	Moves bridge on screen 9.
1210	Moves rope on screens 1 and 13.
1300	Moves drawbridge on screen 10.
1380	Moves all hazards on last screen.
1520 **2120**	Set up screens layout and all variables of hazards used.
2130	Variable dump.
2270	Makes Mr Humpy ring bell and gives bonus.
2370	Graphics characters.
2670	Death of Mr Humpy.
2880	Prints high score table and allows entry of name if score high enough.
3280	Instructions.
3720	Plays tune and gives super bonus when all 15 screens completed.

VARIABLES

X, Y	Mr Humpy's position.
ropeh	If Mr Humpy is on a rope = 1, if not = 0.
rope	Position of swinging rope.
roped	Rope direction.
pole	Position of pole/portcullis.
poled	Pole direction.
ball	Position of rock.
balld	Direction of rock.
na$	Name of high scorer.
hs$	High score.
bonus	Bonus left.
bonusd	Amount by which bonus is decreasing.
screen	Screen number.
bridge	Bridge position.
bridir	Bridge direction.
score	Score.
lives	Lives left.

Da bells, da bells



```
10 REM******************************
20 REM***********DA BELLS***********
30 REM******************************
40 REM*********By A.Chapman********
50 REM******************************
60 REM(C)Computing with the Amstrad
70 DIM na$(10):DIM hs(10)
80 FOR f=1 TO 8:LET na$(f)="*Mr Humpy
*":LET hs(f)=2000-(f*200):NEXT f
90 GOSUB 2370:REM SET UP U.D.G
100 GOSUB 3200:REM INSTRUCTIONS
110 GOSUB 2130:REM SET UP VARIABLES
120 GOSUB 1520:REM SET UP SCREEN
130 PRINT CHR$(22)+CHR$(1):PEN 0:LOCA
TE 18,1:PRINT lives:PRINT CHR$(22)+CH
R$(0):PEN 1
140 REM******************************
150 REM*********MAIN LOOP***********
160 REM******************************
170 ON screen GOSUB 800,1210,960,890,
800,970,960,970,1070,1300,890,1070,12
10,800,1380
180 GOSUB 350
190 GOSUB 230
200 IF x=19 THEN GOTO 2270
210 IF bonus<=1 THEN GOTO 300
220 GOTO 170
230 REM******************************
240 REM*******BONUS COUNT***********
250 REM******************************
260 LOCATE 19-INT(bonus),22:PAPER 1:P
EN 3:PRINT CHR$(225);:PAPER 3:PEN 0:P
RINT CHR$(242)
270 LET bonus=bonus-(1/bonusd)
280 PAPER 0:PEN 1
290 RETURN
300 REM******************************
310 REM********EXPLOSION***********
320 REM******************************
330 FOR f=1 TO 5:SOUND 1,0,10,7,0,0,3
: INK 0,3:BORDER 3:SOUND 1,0,25,5,0,0
,10:INK 0,0:BORDER 0:NEXT f
340 FOR f=1 TO 100:NEXT f:GOTO 2690
350 REM******************************
360 REM*******MOVE HUMPY***********
370 REM******************************
380 IF INKEY(71)=1 AND INKEY(47)=1 AN
D INKEY(63)=1 THEN GOTO 420
390 IF INKEY(47)=0 THEN GOSUB 560:GOT
O 420
400 IF INKEY(63)=0 AND x<19 THEN GOSU
B 480:LET x=x+1:SOUND 1,200,1:a=1
410 IF INKEY(71)=0 AND x>1 THEN GOSUB
 520:LET x=x-1:SOUND 1,200,1:LET a=0
420 IF TEST(32*(x-1)+8,16*(25-(y))+8)
=7 OR TEST(32*(x-1)+16,16*(25-(y))-4)
<>3 OR TEST(32*(x-2)+8,16*(25-(y))+8)
=7 THEN GOTO 2670
430 IF a=1 AND a2=1 THEN PEN 4:LOCATE
 x,y:PRINT CHR$(227):LOCATE x,y-1:PRI
NT CHR$(226):PEN 1:GOTO 450
435 IF a=1 AND a2=2 THEN PEN 4:LOCATE
 x,y:PRINT CHR$(228):LOCATE x,y-1:PRI
NT CHR$(226):PEN 1:GOTO 450
440 IF a=0 AND a2=3 THEN PEN 4:LOCATE
 x,y:PRINT CHR$(247):LOCATE x,y-1:PRI
NT CHR$(249):PEN 1:GOTO 450
445 IF a=0 AND a2=4 THEN PEN 4:LOCATE
 x,y:PRINT CHR$(248):LOCATE x,y-1:PRI
NT CHR$(249):PEN 1
450 IF TEST(32*(x)+8,16*(25-(y))+8)=7
 THEN GOTO 2670
460 IF screen=12 AND (x=4 OR x=18) OR
 screen=15 AND x=16 THEN GOTO 2670
470 RETURN
480 LOCATE x,y:PRINT" ":LOCATE x,y-1:
PRINT" "
490 IF a2=1 OR a2=3 THEN LET a2=2:RET
URN
500 IF a2=2 OR a2=4 THEN LET a2=1
510 RETURN
520 LOCATE x,y:PRINT" ":LOCATE x,y-1:
PRINT" "
530 IF a2=1 OR a2=3 THEN LET a2=4:RET
URN
540 IF a2=2 OR a2=4 THEN LET a2=3
550 RETURN
560 REM******************************
570 REM*******HUMPY JUMP***********
580 REM******************************
590 IF INKEY(47)=0 AND INKEY(71)<>0 A
ND INKEY(63)<>0 THEN a1=1
600 FOR a=-0.75 TO 0.75 STEP 0.5:LOCA
TE  x,y:PRINT" ":LOCATE x,y-1:PRINT"
":LET y=y+3*a
610 IF a=1 AND a1=0 AND x<19 THEN LET
 x=x+1
620 IF a=1 THEN PEN 4:LOCATE x,y:PRIN
T CHR$(228):LOCATE x,y-1:PRINT CHR$(2
26):PEN 1
630 IF a=0 AND a1=0 AND x>1 THEN LET
x=x-1
640 IF a=0 THEN PEN 4:LOCATE x,y:PRIN
T CHR$(248):LOCATE x,y-1:PRINT CHR$(2
49)
650 ON screen GOSUB 800,1210,960,890,
800,970,960,970,1070,1300,890,1070,12
10,800,1380
660 IF (screen=13 OR screen=2) AND ro
peh=0 AND x-1=INT(rope/8) OR (screen=
13 OR screen=2) AND ropeh=0 AND x=INT
(rope/8) THEN GOTO 740
670 IF TEST(32*(x-1)+16,16*(25-(y))-4
)=12 OR TEST(32*(x-2)+16,16*(25-(y))+
8)=7  OR TEST(32*(x-1)+16,16*(25-(y-1
))+8)=12 THEN GOTO 2670
680 IF TEST(32*(x-1)+16,16*(25-(y))+8
)=7 OR TEST(32*(x-1)+16,16*(25-(y))+2
4)=7 THEN GOTO 2670
690 NEXT a
700 LET a1=0
710 LOCATE x,y:PRINT" ":LOCATE x,y-1:
PRINT" "
720 IF TEST(32*(x-1)+16,16*(25-(y))-4
)<>3 OR TEST(32*(x)+16,16*(25-(y))+8)
=7 THEN GOTO 2670
730 RETURN
740 LET ropeh=1:LOCATE x,y:PRINT" ":L
OCATE x,y-1:PRINT" ":LET x=INT(rope/8
):LET y=9
750 LOCATE x,y:PRINT" ":LOCATE x,y-1:
PRINT" ":LET x=INT(rope/32)+5
760 PEN 4:LOCATE x,y-1:PRINT CHR$(226
):LOCATE x,y:PRINT CHR$(229):PEN 1
770 GOSUB 1210
780 IF INKEY(47)=0 THEN GOTO 560
790 GOTO 750
800 REM******************************
810 REM*******MOVE FIRE BALLS******
820 REM******************************
830 LOCATE ball,11:PRINT" ":LOCATE 20
-ball,9:PRINT" "
840 LET ball=ball-1
850 IF ball=1 THEN LET ball=19
860 PEN 7:LOCATE ball,11:PRINT CHR$(2
30):LOCATE 20-ball,9:PRINT CHR$(230)
870 PEN 1
880 FOR f=1 TO 20:NEXT f:RETURN
890 LOCATE ball,9:PRINT" "
900 IF screen=8 AND (ball=6 OR ball=1
1 OR ball=16) THEN PEN 12:LOCATE ball
,9:PRINT CHR$(232):PEN 1
910 LET ball=ball-1
920 IF ball=1 THEN LET ball=19
930 PEN 7:LOCATE ball,9:PRINT CHR$(23
0)
940 IF screen<>8 THEN FOR f=1 TO 20:N
EXT f
950 RETURN
960 FOR f=1 TO 70:NEXT f:RETURN
970 REM******************************
980 REM*****MOVE GUARDS POLES******
990 REM******************************
1000 FOR f=6 TO 16 STEP 5:PEN 12:LOCA
TE f,pole:PRINT CHR$(232):LOCATE f,po
le-1:PRINT" ":NEXT f
1010 LET pole=pole+poled
1020 IF pole=5 THEN LET poled=1
1030 IF pole=11 THEN LET poled=-1
1040 PEN 1
1050 IF screen=8 THEN GOSUB 890
1060 RETURN
1070 REM******************************
1080 REM*******MOVE BRIDGE**********
1090 REM******************************
1100 PAPER 1 :PEN 3:LOCATE bridge,12:
PRINT CHR$(225);:PAPER 0:PRINT" ":LOC
```

```
ATE bridge-4,12:PRINT" ";:PAPER 1:PRI
NT CHR$(225):PAPER 0:PEN 1
1110 LET bridge=bridge+bridir
1120 IF bridge=9 THEN LET bridir=1
1130 IF bridge=13 THEN LET bridir=-1
1140 IF screen=9 THEN GOTO 1190
1150 LOCATE 4,pole:PEN 12:PRINT CHR$(
232):LOCATE 17,pole:PRINT CHR$(232):P
EN 1:LOCATE 4,pole-1:PRINT" ":LOCATE
17,pole-1:PRINT" "
1160 LET pole=pole+poled
1170 IF pole=7 THEN LET poled=1
1180 IF pole=11 THEN LET poled=-1
1190 PAPER 0:PEN 1
1200 RETURN
1210 REM*****************************
1220 REM*********MOVE ROPE**********
1230 REM*****************************
1240 PLOT 270,368:DRAWR ROPE-128,-120
,0
1250 LET ROPE=ROPE+ROPED
1260 IF ROPE>225 THEN LET ROPED=-8
1270 IF ROPE<40 THEN LET ROPED=8
1280 PLOT 270,368:DRAWR ROPE-128,-120
,10
1290 RETURN
1300 REM*****************************
1310 REM********MOVE DRAWBRIDGE*****
1320 REM*****************************
1330 PAPER 1:PEN 3:LOCATE bridge,12:P
RINT CHR$(225):LOCATE 19-bridge,12:PR
INT CHR$(225):PAPER 0:LOCATE bridge+1
,12:PRINT" ":LOCATE 18-bridge,12:PRIN
T" "
1340 LET bridge=bridge+bridir:IF brid
ge=8 THEN LET bridir=-bridir
1350 IF bridge=5 THEN LET bridir=1
1360 PEN 1
1370 RETURN
1380 REM*****************************
1390 REM********LAST SCREEN********
1400 REM*****************************
1410 LOCATE 5,pole:PEN 12:PRINT CHR$(
232):LOCATE 10,pole-3:PRINT CHR$(149)
;CHR$(149);CHR$(149):LOCATE 10,pole-2
:PRINT CHR$(231);CHR$(231);CHR$(231):
LOCATE 5,pole-1:PRINT" ":LOCATE 10,po
le-1:PRINT"   "
1420 LOCATE 16,pole-1:PRINT CHR$(232)
:LOCATE 16,pole-2:PRINT" "
1430 LET pole=pole+poled
1440 IF pole=6 THEN LET poled=1
1450 IF pole=12 THEN LET poled=-1
1460 LOCATE ball,11:PRINT" "
1470 LET ball=ball+balld
1480 IF ball=15 THEN LET balld=-1
1490 IF ball=7 THEN LET balld=1
1500 LOCATE ball,11:PEN 7:PRINT CHR$(
230)
1510 PEN 1:RETURN
1520 REM*****************************
1530 REM********SCREEN SET UP********
1540 REM*****************************
1550 MODE 0:IF screen=16 THEN screen=
1
1560 ON screen GOSUB 1580,1950,1720,1
720,1720,1770,2080,1770,1900,1860,208
0,1900,1950,2080,2010
1570 RETURN
1580 FOR f=12 TO 24
1590 PAPER 1:PEN 3
1600 LOCATE 1,f:PRINT STRING$(20,CHR$
(225))
1610 NEXT f
1620 FOR f=1 TO 2:LOCATE 1,f:PRINT ST
RING$(20,CHR$(225)):NEXT f
1630 PAPER 0:PEN 1
1640 LET ball=19
1650 LOCATE 20,3:PEN 13:PRINT CHR$(23
5):LOCATE 20,4:PRINT CHR$(236):FOR f=
5 TO 10:LOCATE 20,f:PRINT CHR$(237):N
EXT f:PEN 1
1660 PRINT CHR$(22)+CHR$(1)
1670 LOCATE 19,22:PAPER 1:PEN 0:PRINT
 CHR$(238);CHR$(239):LOCATE 19,23:PRI
NT CHR$(240);CHR$(241):LOCATE 1,22:PR
INT STRING$(18,CHR$(154)):PAPER 0:PEN
 1
1680 LOCATE 2,1:PEN 0:PRINT"1UP:";sco
re:LOCATE 12,1:PRINT"LIVES:":PEN 1
1690 PRINT CHR$(22)+CHR$(0)
1700 LOCATE 2,23:PAPER 4:PEN 0:PRINT"
SCREEN";screen:LOCATE 14,23:PRINT"BON
US":PAPER 0:PEN 1
1710 RETURN
1720 GOSUB 1580
1730 FOR f=5 TO 15 STEP 5
1740 LOCATE f,12:PRINT"  ":LOCATE f,1
3:PRINT"  ":LOCATE f,14:PRINT"  ":NEX
T f
1750 FOR f=7 TO 17 STEP 5:PEN 2:LOCAT
E f,12:PRINT CHR$(214):LOCATE f,13:PR
INT CHR$(143):LOCATE f,14:PRINT CHR$(
143):PEN 1:NEXT f
1760 RETURN
1770 GOSUB 1720
1780 FOR f=6 TO 16 STEP 5:PEN 10:LOCA
TE f,13:PRINT CHR$(233):LOCATE f,14:P
RINT CHR$(234):NEXT f
1790 PEN 1
1800 pole=12:poled=-1
1810 RETURN
1820 GOSUB 1580
1830 FOR f=12 TO 20:LOCATE 5,f:PRINT
STRING$(11," "):NEXT f
1840 FOR f=12 TO 20:LOCATE 15,f:PEN 2
:PRINT CHR$(143):NEXT f:LOCATE 15,12:
PRINT CHR$(214):PEN 1
1850 RETURN
1860 GOSUB 1820
1870 PAPER 1:PEN 3:LOCATE 15,12:PRINT
 CHR$(225):PEN 1:PAPER 0
1880 LET bridge=5:LET bridir=1
1890 RETURN
1900 GOSUB 1820
1910 LET bridge=13:LET bridir=-1
1920 LET ball=19
1930 LET pole=11:LET poled=-1
1940 RETURN
1950 GOSUB 1820
1960 LET rope=128:LET roped=8:LET rop
eh=0
1970 LOCATE 16,11:PAPER 1:PEN 3:PRINT
 STRING$(5,CHR$(225)):LOCATE 16,10:PR
INT STRING$(5,CHR$(225)):PAPER 0:PEN
1
1980 LOCATE 15,11:PEN 2:PRINT CHR$(14
3):LOCATE 15,10:PRINT CHR$(214)
1990 LOCATE 15,12:PRINT CHR$(143):PEN
 1
2000 RETURN
2010 GOSUB 1580
2020 LOCATE 4,12:PRINT"  ":LOCATE 4,1
3:PRINT"  ":LOCATE 4,14:PRINT"  ":PEN
 2:LOCATE 6,12:PRINT CHR$(214):FOR f=
13 TO 14:LOCATE 6,f:PRINT CHR$(143):N
EXT f:PEN 1
2030 LOCATE 17,5:PAPER 1:PEN 3:PRINT
STRING$(3,CHR$(225)):LOCATE 18,6:PRIN
T CHR$(225);CHR$(225):LOCATE 9,3:PRIN
T STRING$(5,CHR$(225)):LOCATE 9,4:PRI
NT STRING$(5,CHR$(225)):PAPER 0
2040 LOCATE 18,3:PEN 11:PRINT CHR$(25
4):LOCATE 18,4:PRINT CHR$(255):PEN 1
2050 LET ball=12:LET balld=1:LET pole
=12:LET poled=-1
2060 PEN 10:LOCATE 5,13:PRINT CHR$(23
3):LOCATE 5,14:PRINT CHR$(234):PEN 1
2070 FOR f=4 TO 10:PEN 12:LOCATE 10,f
:PRINT CHR$(149);CHR$(149);CHR$(149):
NEXT f:PEN 1:RETURN
2080 GOSUB 1820
2090 LOCATE 8,12:PAPER 1:PEN 3:PRINT
CHR$(225):LOCATE 12,12:PRINT CHR$(225
)
2100 PRINT CHR$(22)+CHR$(1):LOCATE 8,
3:PEN 13:PRINT CHR$(235);"    ";CHR$(
235):LOCATE 8,4:PRINT CHR$(236);"
";CHR$(236):LOCATE 8,5:PRINT CHR$(237
);"    ";CHR$(237):LOCATE 8,6:PRINT C
HR$(237);"    ";CHR$(237)
2110 PRINT CHR$(22)+CHR$(0):PAPER 0:P
EN 1
2120 RETURN
2130 REM*****************************
2140 REM********VARIABLE DUMP*******
2150 REM*****************************
```

```
2170 LET y=11:LET x=1
2180 screen=VAL(chose$)
2190 PEN 1
2200 LET ropeh=0
2210 LET bonus=10:LET bonusd=5
2220 LET lives=3:LET n=1
2230 LET score=0:LET n2=1
2240 ENV 6,15,-1,1:ENT 1,30,10,1:ENV
5,15,-1,10
2250 ENV 1,10,-1,2:ENV 2,15,1,3:ENV 3
,1,0,2,15,-1,20
2260 RETURN
2270 REM*****************************
2280 REM******REACH END OF SCREEN*****
2290 REM*****************************
2300 FOR f=1 TO 4:PEN 4:LOCATE x,y-1:
PRINT CHR$(226):LOCATE x,y:PRINT CHR$
(229):LOCATE x,y-2:PRINT" ":SOUND 1,1
40,12,0,1:SOUND 1,120,12,0,1:LOCATE x
,y:PRINT" ":LOCATE x,y-1:PRINT CHR$(2
29):LOCATE x,y-2:PRINT CHR$(226)
2310 SOUND 1,120,12,0,1:SOUND 1,140,1
2,0,1:NEXT f
2320 LET score=score+10*INT(bonus)
2330 LET screen=screen+1
2340 LET x=1:LET y=11:LET bonus=10
2350 IF screen=16 THEN GOSUB 3720
2360 GOTO 120
2370 REM*****************************
2380 REM***********U.D.G*************
2390 REM*****************************
2400 SYMBOL AFTER 224
2410 SYMBOL 225,254,254,254,0,239,239
,0
2420 SYMBOL 226,56,124,126,244,236,19
4,68,120
2430 SYMBOL 227,60,254,254,62,60,24,2
4,30
2440 SYMBOL 228,60,254,254,126,60,109
,199,102
2450 SYMBOL 229,250,255,126,56,56,29,
15,7
2460 SYMBOL 254,0,60,46,70,70,39,56,5
5
2470 SYMBOL 255,250,60,60,126,126,126
,255,36
2480 SYMBOL 230,24,62,126,255,255,126
,126,40
2490 SYMBOL 231,24,24,24,24,24,126,60
,24
2500 SYMBOL 232,24,24,60,90,24,60,90,
24
2510 SYMBOL 233,60,66,153,129,165,231
,126,60
2520 SYMBOL 234,24,126,231,165,36,36,
36,231
2530 SYMBOL 235,0,0,24,60,66,102,90,1
02
2540 SYMBOL 236,90,129,255,88,152,188
,152,128
2550 SYMBOL 237,128,128,128,128,128,1
28,128,128
2560 SYMBOL 238,31,63,63,223,192,92,7
2,74
2570 SYMBOL 239,248,252,252,250,2,2,2
,66
2580 SYMBOL 240,75,66,66,64,64,64,32,
31
2590 SYMBOL 241,66,222,74,10,10,10,4,
248
2600 SYMBOL 242,132,74,40,151,95,40,6
8,146
2610 SYMBOL 243,0,16,45,103,114,125,2
8,0
2620 SYMBOL 244,0,2,226,254,254,224,0
,0
2630 SYMBOL 247,60,127,127,126,60,102
,227,102
2640 SYMBOL 248,60,127,127,127,60,24,
24,120
2650 SYMBOL 249,28,62,126,47,55,67,34
,30
2655 SYMBOL 250,198,165,198,165,6,40,
40,16
2660 RETURN
2670 REM*****************************
2680 REM*****************************
2690 REM*****************************
2700 REM******DEATH SEQUENCE******
2710 REM*****************************
2720 LET x=x+1
2730 LOCATE x-1,y-1:PRINT"  ":LOCATE
x-1,y:PEN 15:PRINT CHR$(243);CHR$(244
)
2740 PEN 1
2750 IF bonus<=1 THEN FOR f=1 TO 500:
NEXT f:GOTO 2810
2760 RESTORE 2760
2770 FOR f=1 TO 10
2780 READ n,d,n1,d1,n2,d2
2790 SOUND 1,n,d,7:SOUND 2,n1,d1,6:SO
UND 4,n2,d2,4
2800 NEXT f
2810 LET lives=lives-1
2820 INK 2,20
2830 IF lives=0 THEN GOTO 2880
2840 LET bonus=10:LET x=1:LET y=11
2850 GOTO 120
2860 DATA 1276,100,319,100,159,100,11
36,20,284,20,142,20,1073,50,268,50,13
4,50,1276,20,319,20,159,20,0,100,0,10
0,0,100
2870 DATA 1276,100,319,100,159,100,11
36,20,284,20,142,20,1073,50,268,50,13
4,50,1276,70,319,70,156,70,902,100,22
5,100,113,100
2880 REM*****************************
2890 REM*********HIGH SCORE*********
2900 REM*****************************
2910 MODE 1
2920 LOCATE 15,5:PEN 3:PRINT"HIGH SCO
RE"
2930 PEN 2
2940 LOCATE 4,6:PRINT CHR$(150);:PRIN
T STRING$(30,CHR$(154));:PRINT CHR$(1
56)
2950 FOR f=7 TO 17:LOCATE 4,f:PRINT C
HR$(149):LOCATE 35,f:PRINT CHR$(149):
NEXT f
2960 LOCATE 4,18:PRINT CHR$(147);:PRI
NT STRING$(30,CHR$(154));:PRINT CHR$(
153)
2970 FOR f=1 TO 8
2980 IF score>hs(f) THEN GOTO 3090
2990 NEXT f
3000 FOR f=1 TO 8:PEN 1:LOCATE 8,f+8:
PRINT na$(f):LOCATE 18,f+8:PEN 3:PRIN
T".....";hs(f):NEXT f
3010 LOCATE 1,19:PRINT STRING$(120,"
")
3020 IF INKEY$<>"" THEN GOTO 3020
3030 PEN 2:LOCATE 1,22:PRINT STRING$(
40,CHR$(154)):LOCATE 1,24:PRINT STRIN
G$(40,CHR$(154)):PEN 1:LOCATE 6,23:IN
PUT"SELECT START SCREEN (1-15)";chose
$
3040 IF VAL(chose$)>15 OR VAL(chose$)
<1 THEN LOCATE 1,23:PRINT STRING$(40,
" "):GOTO 3030
3050 LOCATE 1,23:PRINT STRING$(40," "
)
3060 LOCATE 11,23:PEN 3:PRINT"PRESS";
:PEN 1:PRINT"< SPACE >";:PEN 3:PRINT"
TO PLAY.":PEN 1
3070 IF INKEY(47)<>0 THEN GOTO 3070
3080 GOTO 110
3090 LET a$="ABCDEFGHIJKLMNOPQRSTUVWX
YZ .#%&()!{}?*+"+CHR$(250)
3100 LET c=19:LOCATE 1,22:PEN 1:PRINT
 a$
3110 LOCATE 2,19:PEN 3:PRINT"USE CURS
OR KEY'S LEFT,RIGHT AND COPY":LOCATE
3,20:PRINT"TO SELECT LETTERS.(MAXIMUM
 OF 10.)":PEN 2:LOCATE 1,21:PRINT STR
ING$(40,CHR$(154))
3120 LET x$=""
3130 FOR z=1 TO 10
3140 LOCATE c,23:PEN 2:PRINT" "
3150 IF INKEY(1)=0 AND c<40 THEN LET
c=c+1
3160 IF INKEY(8)=0 AND c>1 THEN LET c
=c-1
3165 IF INKEY(9)=0 AND c=40 THEN LOCA
TE 7,f+8:PRINT"          ":LET x$=""
:GOTO 3120
3170 IF INKEY(9)=0 THEN LET x$=x$+MID
$(a$,c,1):LOCATE 7+z,f+8:PEN 1:PRINT
```


```
MID$(a$,c,1):FOR a=1 TO 200:NEXT a:NE
XT z:GOTO 3210
3180 LOCATE c,23:PRINT"*"
3190 FOR a=1 TO 50:NEXT a
3200 GOTO 3140
3210 LET hs(8)=score:LET na$(8)=x$
3220 LET f=0
3230 FOR z=1 TO 7
3240 IF hs(z)<hs(z+1) THEN LET t=hs(z
+1):LET hs(z+1)=hs(z):LET hs(z)=t:LET
 a$=na$(z+1):LET na$(z+1)=na$(z):LET
na$(z)=a$:LET f=1
3250 NEXT z
3260 IF f=1 THEN GOTO 3220
3270 GOTO 3000
3280 REM**************************
3290 REM*******INSTRUCTIONS*******
3300 REM**************************
3310 MODE 1:INK 0,0:BORDER 0
3320 PAPER 1:PEN 3:PRINT STRING$(40,C
HR$(225)):FOR f=1 TO 7:LOCATE 1,f:PRI
NT CHR$(225):LOCATE 40,f:PRINT CHR$(2
25):NEXT f:PAPER 0
3330 LOCATE 17,3:PEN 2:PRINT STRING$(
8,CHR$(131)):LOCATE 17,4:PEN 3:PRINT"
DA BELLS":PEN 1:LOCATE 17,5:PRINT STR
ING$(8,CHR$(140))
3340 PAPER 1:PEN 3:PRINT:PRINT STRING
$(80,CHR$(225)):PAPER 0
3350 PEN 2
3360 PRINT"  Mr Humpy is very sad.The
 love of his  life Esmerelda has been
 captured and putin the horrible Bost
on-de-Stump."
3370 PRINT"  It is your task to help
him run along the heavily guarded wal
l of the Stump  and free Esmerelda f
rom her cell."
3380 PRINT"  There are however some o
bstacles whichare out to stop you. Th
ese include manicFrenchmen, magenta-c
oloured rocks and a barrel of TNT whi
ch is about to explode."
3390 PAPER 1:PEN 3:PRINT STRING$(40,C
HR$(225)):PAPER 0
3400 LOCATE 1,22:PEN 2:PRINT STRING$(
40,CHR$(131)):LOCATE 1,24:PRINT STRIN
G$(40,CHR$(140)):LOCATE 7,23:PEN 3:PR
INT"Press <SPACE BAR> to continue"
3410 IF INKEY(47)<>0 THEN GOTO 3410
3420 CLS
3430 PAPER 1:PEN 3:PRINT STRING$(40,C
HR$(225)):FOR f=1 TO 7:LOCATE 1,f:PRI
NT CHR$(225):LOCATE 40,f:PRINT CHR$(2
25):NEXT f:PAPER 0
3440 LOCATE 17,3:PEN 2:PRINT STRING$(
8,CHR$(131)):LOCATE 17,4:PEN 3:PRINT"
DA BELLS":PEN 1:LOCATE 17,5:PRINT STR
ING$(8,CHR$(140))
3450 PRINT:PEN 3:PAPER 1:PRINT STRING
$(80,CHR$(225)):PEN 2:PAPER 0
3460 PRINT"  To complete each screen
you must ring the bell. If you can ri
ng it before the TNT explodes you wil
l be given a bonus. However if the TN
T explodes one of your lives will be
lost."
3470 PRINT"  Once you have finished a
ll 15 screens a Super Bonus is given
and you  have to start again. But thi
s time the  fuse on the TNT will be b
urning a lot faster."
3480 PAPER 1:PEN 3:PRINT:PRINT STRING
$(80,CHR$(225)):PEN 1:PAPER 0
3490 LOCATE 1,22:PEN 2:PRINT STRING$(
40,CHR$(131)):LOCATE 1,24:PRINT STRIN
G$(40,CHR$(140))
3500 LOCATE 7,23:PEN 3:PRINT"Press <S
PACE BAR> to continue"
3510 IF INKEY$="" THEN GOTO 3510
3520 IF INKEY(47)<>0 THEN GOTO 3520
3530 IF INKEY$<>"" THEN GOTO 3530
3540 CLS
3550 PAPER 1:PEN 3:PRINT STRING$(40,C
HR$(225)):FOR f=1 TO 7:LOCATE 1,f:PRI
NT CHR$(225):LOCATE 40,f:PRINT CHR$(2
25):NEXT f:PAPER 0
3560 LOCATE 17,3:PEN 2:PRINT STRING$(
8,CHR$(131)):LOCATE 17,4:PEN 3:PRINT"
DA BELLS":PEN 1:LOCATE 17,5:PRINT STR
ING$(8,CHR$(140))
3570 PAPER 1:PEN 3:PRINT:PRINT STRING
$(80,CHR$(225)):PAPER 0
3580 PEN 3
3590 LOCATE 17,10:PRINT"THE KEYS":PEN
 2:LOCATE 17,11:PRINT STRING$(8,CHR$(
131))
3600 PEN 3:LOCATE 14,12:PRINT"X....mo
ve left"
3610 PEN 1:LOCATE 14,14:PRINT"Z....mo
ve right"
3620 PEN 2:LOCATE 14,16:PRINT"SPACE..
.to jump"
3630 PEN 1:LOCATE 9,18:PRINT"Z and SP
ACE to jump left"
3640 PEN 3:LOCATE 9,20:PRINT"X and SP
ACE to jump right"
3670 PEN 2:LOCATE 1,22:PRINT STRING$(
40,CHR$(154)):LOCATE 1,24:PRINT STRIN
G$(40,CHR$(154)):PEN 3:LOCATE 8,23:IN
PUT"SELECT START SCREEN (1-15)";chose
$
3680 IF VAL(chose$)<1 OR VAL(chose$)>
15 THEN LOCATE 1,23:PRINT STRING$(40,
" "):GOTO 3670
3690 LOCATE 1,23:PRINT STRING$(40," "
):LOCATE 11,23:PEN 2:PRINT"Press <SPA
CE> to Play."
3700 IF INKEY$<>" " THEN GOTO 3700
3710 RETURN
3720 REM**************************
3730 REM******CONGRATULATIONS******
3740 REM**************************
3750 LET bon=INT(RND*40)*10
3760 LOCATE 6,2:PEN 14:PRINT bon+500;
"BONUS "
3770 RESTORE 3830
3780 FOR F=1 TO 24:READ N:SOUND 4,N,2
0,15,1:NEXT F
3790 FOR F=1 TO 5000:NEXT F
3800 LET score=score+500+bon
3810 PEN 1:LET bonusd=bonusd-1:IF bon
usd=0 THEN LET bonusd=1
3820 RETURN
3830 DATA 60,53,47,45,60,0,45,47,45,4
0,53,0,53,47,45,36,40,40,45,45,47,53,
47,60,9999,9999
```

Give your fingers a rest . . .

All the listings from this month's issue are available on cassette.

See our special offer on Page 77.

ROLAND WADDILOVE says . . .

It's so easy from this

ONE of the major differences between a tape and a disc system is the amazing speed at which programs load and save. What took minutes before now takes seconds.

Not only does this great speed advantage enable you to use your time more efficiently, and generally make programming more bearable, but it allows programs to be written which were impractical before. The long wait for programs or data to load has been banished.

A menu program to display the contents of a disc and allow the easy selection of programs demonstrates the advantages of the system admirably.

Most disc menu programs have the names of all the files stored in data statements. They then read the names, print a list and allow you to select a program.

This is fine if you don't intend to change the contents of the disc very often, but a pain in the neck if you do, and all the data statements have to be changed when it is used with another disc.

What is needed is an intelligent menu – one which looks at the disc to see what is on it rather than fixed data statements. Disc Menu – I know it's not an original title, but it is descriptive – is such a program. It does not rely on data and it does not matter what is on the disc, or whether it has been changed in any way by adding, deleting or renaming files since the menu was added.

In fact you could load Disc Menu off one disc, insert another with a completely different set of files and run it with this.

Simply:

```
RUN
```

or

```
CHAIN "menu"
```

(after typing it in and saving it first of course), and a menu of all the files will be displayed with a pointer next to the first.

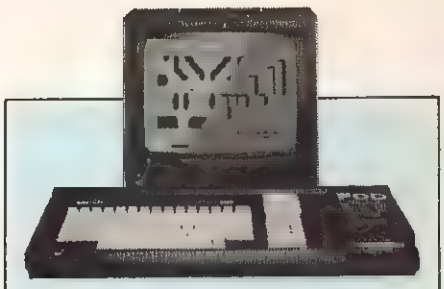
The pointer, indicating the current selection, can be moved up and down the list using the cursor keys and a file loaded and run by pressing Enter. It couldn't be easier to use, and it makes the whole system so much friendlier.

There is also an option to selectively list a particular type of file, such as all the Basic programs or all the binary files, and you could extend the list to include other types if desired.

Binary files present a few problems – you can't load them below HIMEM, and are they machine code programs or data such as a screen dump? So if the current selection is a binary file, detected by looking for .BIN in the name, then you are requested to input a new value for HIMEM (just press

```
10 REM Disc Menu
20 REM By R.A.Waddilove
30 REM (c)Computing With The Amstrad
40 REM ----------------------------
50 GOSUB 180:REM Initialise
60 GOSUB 460:REM get names
70 IF INSTR(file$,".BIN")=0 THEN RUN
file$
80 CLS:INPUT "HIMEM ";a
90 IF a=&8000 THEN MEMORY &7FFF ELSE
IF a THEN MEMORY a-1
100 PRINT "Load or Run (L/R) ?"
110 k$=""
120 WHILE k$<>"L" AND k$<>"R"
130 k$=UPPER$(INKEY$)
140 WEND
150 IF k$="L" THEN LOAD file$ ELSE RU
N file$
160 END
170 REM --------------------------
180 REM Initialise
190 !DISC
200 MODE 1:INK 0,0:BORDER 0
210 DEFINT b-z
220 DIM name$(64),coord(64,1)
230 FOR i=1 TO 20
240 READ a$
250 POKE &8000+i,VAL("&"+a$)
260 NEXT
270 DATA DD,6E,00,DD,66,02,CD,75,BB,C
D,60,BB,DD,6E,04,DD,66,05,77,C9
280 readchar=&8001:char=0
290 prog=1:type$=""
300 PAPER 3:LOCATE 9,5
310 PRINT " Print menu for...? "
320 PAPER 0:PEN 2
330 LOCATE 12,10:PRINT "1. All files"
340 LOCATE 12,12:PRINT "2. Basic only
"
350 LOCATE 12,14:PRINT "3. Binary onl
y"
360 LOCATE 12,16:PRINT "4. Back up on
ly"
370 WHILE type$<"1" OR type$>"4"
380 type$=INKEY$
390 WEND
400 ON VAL(type$) GOTO 410,420,430,44
0
410 type$="":RETURN
420 type$=".BAS":RETURN
430 type$=".BIN":RETURN
440 type$=".BAK":RETURN
450 REM --------------------------
460 REM Get program names
470 MODE 2:INK 1,0:PEN 1:CAT
480 FOR x=1 TO 61 STEP 20
490 y=4:CALL readchar,@char,x,y
500 WHILE char<>32
510 name$(prog)=""
520 FOR i=0 TO 11
530 CALL readchar,@char,x+i,y
540 name$(prog)=name$(prog)+CHR$(char
)
```

to choose menu...



This program is suitable for the new CPC664 (above), which has built-in disc drives, as well as the separate disc drive used with the CPC464.

Enter if you don't want to change it). Then you must say whether it is to be loaded or run.

How does Disc Menu work? At first I thought it would be impossible given the scant information in the manual supplied with the disc drive. It is in fact very simple.

If you wanted to find out what is on a disc what would you do? Type CAT and read the names printed on the screen. This is exactly what the program does.

Although there isn't a Basic command to read the screen, the operating system is capable of it. To use the routine CALL &BB60, the code for the character at the text cursor position is in the A register on return. A very short (20 byte) machine code routine was written to read the character at any position and place the code in a variable. The disc is catalogued and the names read into an array, provided they are of the right type, by scanning the screen using the readchar machine code routine.

You will not see this happening as the current pen ink is set to black, the same as the background. It doesn't fool the readchar routine though. What you will notice is a slight delay while it carries this out.

It is then a relatively simple matter to list the names and print the pointer. So it does not matter what is on the disc, it is simply catalogued and the information read off the screen. A menu is prepared from this.

Please note that there is only room for about 40 file names on the screen. If you have more than this on the disc then choose any option other than 1, which prints all the files.

180-290	Initialise variables.
300-440	Get type – add your own if desired.
460-590	Read names into array.
640-750	Print menu.
760-880	Select program.
name$(64)	File names.
coord(64,1)	Print positions.
prog	Number of programs.
type$	File type.
char	Character read from screen.

```
550 NEXT
560 IF INSTR(name$(prog),type$) THEN
prog=prog+1
570 y=y+1:CALL readchar,@char,x,y
580 WEND
590 NEXT
600 MODE 1:PEN 1:INK 1,24
610 prog=prog-1

620 IF prog=0 THEN PRINT TAB(5)"No fi
les of that type on disc...":FOR i=0
TO 5000:NEXT:RUN
630 PAPER 3
640 LOCATE 10,1:PRINT " D I S C    M
E N U "
650 PAPER 0
660 y=3:num=(prog\2)+(1 AND prog)
670 FOR i=1 TO prog

680 y=y+1-1*(prog<21):IF i=num+1 THEN
 y=4-1*(prog<21)
690 coord(i,0)=4+(20 AND i>num):coord
(i,1)=y
700 LOCATE coord(i,0),coord(i,1)
710 PEN 2:PRINT LEFT$(name$(i),INSTR(
name$(i),".")-1);
720 PEN 3:PRINT MID$(name$(i),INSTR(n
ame$(i),"."))
730 NEXT
740 LOCATE 4,25:PEN 3:PAPER 1

750 PRINT " Select with ";CHR$(240);"
 and ";CHR$(241);" then ENTER ";
760 i=1:PEN 1:PAPER 0
770 GOSUB 870
780 WHILE INKEY$<>CHR$(13)

790 IF INKEY(0)>-1 THEN GOSUB 850:i=i
+(i>1):GOSUB 870
800 IF INKEY(2)>-1 THEN GOSUB 850:i=i
-(i<prog):GOSUB 870
810 FOR j=1 TO 200:NEXT
820 WEND
830 file$=name$(i)
840 RETURN
850 LOCATE coord(i,0)-2,coord(i,1):PR
INT " ";STRING$(14,CHR$(9));" "
860 RETURN
870 LOCATE coord(i,0)-2,coord(i,1):PR
INT CHR$(243);STRING$(14,CHR$(9));CHR
$(242)
880 RETURN
```

Give your fingers a rest . . .

All the listings from this month's issue are available on cassette.

See our special offer on Page 77.

OUR sixth reference chart puts all the opcodes and firmware routines from Mike Bibby's machine code articles together in handy form.

LD B,n	06 n
LD C,n	0E n
LD D,n	16 n
LD E,n	1E n
LD H,n	26 n
LD L,n	2E n
LD A,n	3E n

Table I: LD r,n opcodes

r	r' B	C	D	E	H	L	A
B	40	41	42	43	44	45	47
C	48	49	4A	4B	4C	4D	4F
D	50	51	52	53	54	55	57
E	58	59	5A	5B	5C	5D	5F
H	60	61	62	63	64	65	67
L	68	69	6A	6B	6C	6D	6F
A	78	79	7A	7B	7C	7D	7F

Table II: Opcodes for LD r,r'

r	INC r	DEC r
B	04	05
C	0C	0D
D	14	15
E	1C	1D
H	24	25
L	2C	2D
A	3C	3D

Table III: INC r and DEC r opcodes

r	ADD A,r	SUB r
B	80	90
C	81	91
D	82	92
E	83	93
H	84	94
L	85	95
A	87	97

Table IV: ADD A,r and SUB r opcodes

mnemonic	opcode
CALL pq	CD q p
RET pq	C9
LD A,(pq)	3A q p
LD (pq),A	32 q p
ADD A,n	C6 n
SUB n	D6 n

Table V: Additional opcodes

mnemonics	opcodes
LD BC,mn	01 n m
LD DE,mn	11 n m
LD HL,mn	21 n m

Table VI: Opcodes for loading register pairs with constants

mnemonics	opcodes
LD (pq),BC	ED 43 q p
LD (pq),DE	ED 53 q p
LD (pq),HL	ED 63 q p
LD (pq),HL	22 q p

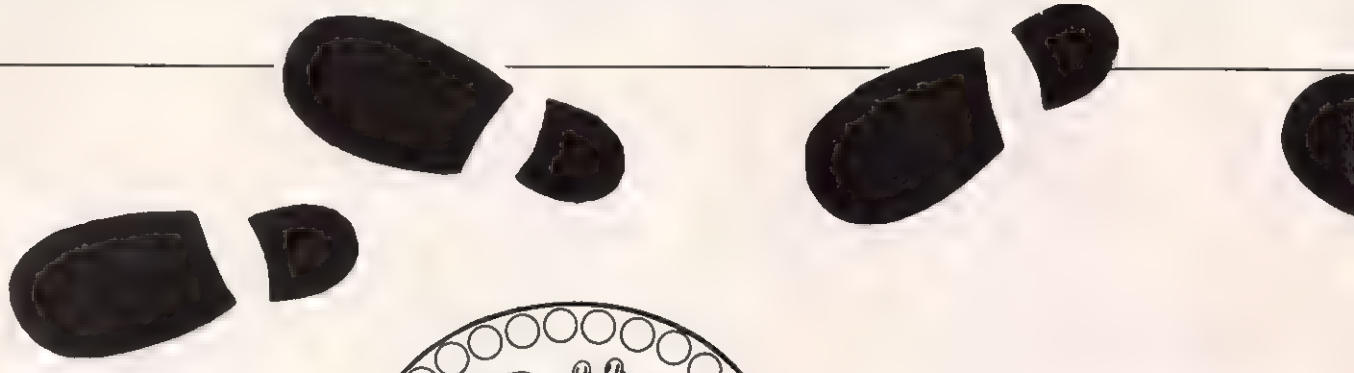
Table VII: Opcodes for 16 bit pokes

mnemonics	opcodes
LD BC,(pq)	ED 4B q p
LD DE,(pq)	ED 5B q p
LD HL,(pq)	ED 5B q p
LD HL,(pq)	2A q p

Table VIII: 16 bit peeks with register pairs

name	official name	address	description
CharIn	KM Wait Key	&BB18	Waits for the next key pressed. On exit, Carry is set and A contains the characters. The registers are preserved, the flags corrupt.
CharOut	Txt Output	&BB5A	Outputs character in A to screen obeying control codes. Everything's preserved.
ClrGraf	Gra Clear Window	&BBDB	Clear the graphics window On exit AF,BC,DE and HL are corrupt.
ClrText	Txt Clear Window	&BB6C	Clears currently selected stream to its paper ink. AF,BC,DE and HL are corrupted.
GrafLine	Gra Line Absolute	&BBF6	Draws a line from current graphics position to graphics position specified. DE contains destination's X coordinate. HL contains destination's Y coordinate. AF,BC,DE and HL corrupt. All other registers are preserved.
PosTCur	Txt Set Cursor	&BB75	Moves text cursor to specified position. On entry H has column number, L has row number. On exit AF and HL are corrupt.
SetPaper	Txt Set Paper	&BB96	Sets paper ink for current screen. On entry A contains the ink number. On exit AF and HL are corrupt.
TextWin	Txt Win Enable	&BB66	Sets size of a text window for the current stream. On entry H,D contain the column numbers of the window. L,E contain the row numbers of the window. On exit AF,BC,DE,HL are corrupt.

Table IX: Useful firmware routines



ALAN McLACHLAN delves deeper into debugging

EVENIN' all. I've already had some rather nice letters from people who have benefited from the debugging tips we've looked at so far.

One delighted chap said that he'd given up the ghost on eight of his listings but has since managed to resurrect them, all thanks to my hints. I was quite chuffed and it encouraged me to seek additional ways to help people with listing problems,

In the last issue of *Computing with the Amstrad* we looked at methods of moving progressively through your program using the STOP command to narrow down the location of an error line.

There are a couple more similar ideas that you could use in circumstances where you don't want the "Break at line..." message printed on the screen.

The command END inserted in the same way as the STOP command will result in only the "Ready" prompt being printed.

If you want to avoid any message at all printed on screen the method to use is to insert a "dummy" line in the form:

```
20 GOTO 20
```

This literally suspends the program at line 20 until you press Escape twice. I used this last month to find a well hidden mistake in my young son's Smiley program. If I take you briefly through the procedure it may help you if you encounter similar problems.

Once the screen had initialised, but before the game got under way, the maze scrolled up one character, and the resulting screen display was just about recognisable, but rubbish.

From a quick read through the listing the appropriate subroutine appeared to be typed in correctly. I mentioned last month the advantage of being able to read through any part of a listing and know what was going on.

Here I didn't have to read through the whole program because I was able to find the subroutine that initialised the screen. This was made easier by reading the list of subroutines published with the program.

I first tried the STOP hint that I showed you. This, of course, printed the "Break at line..." message and scrolled the screen anyway, so it didn't help in this case. The way I found the bug was to type in the dummy line:

```
905 GOTO 905
```

and check to see if the scroll had occurred. It hadn't, so I deleted line 905 and replaced it with:

```
915 GOTO 915
```

and so on until I had used line 945 and knew that I was just after the instant that the screen scrolled. I re-checked by putting in the dummy line again just before to make sure.

To cut a long story short, I then listed what appeared to be the offending line and found in fact two lines 930 and 940 joined together as one.

How could this happen? Very easily when you're using the Copy key for editing. I'm sure you know what I mean, but I'll show you an example. If:

```
LIST 20
```

results in the following being displayed:

```
20 MODE 1:GOSUB 500:GOSUB 600:GOSUB
700
30 PRINT "A1"
```

line 30 must have been copied onto the end of line 20, otherwise it could not be listed with List 20. This won't happen often, but it has happened to me more than once, and in the way of things I've no doubt it will happen again. It's certainly worth bearing in mind.

Another useful hint for finding an error without destroying any part of your program is to REM the particular line that you think is causing the trouble.

As you should know by now, anything following a REM statement (short for REMark or REMinder if you like) is ignored by the program (sometimes when I'm talking to the kids I get the feeling that everything I say begins with a REM).

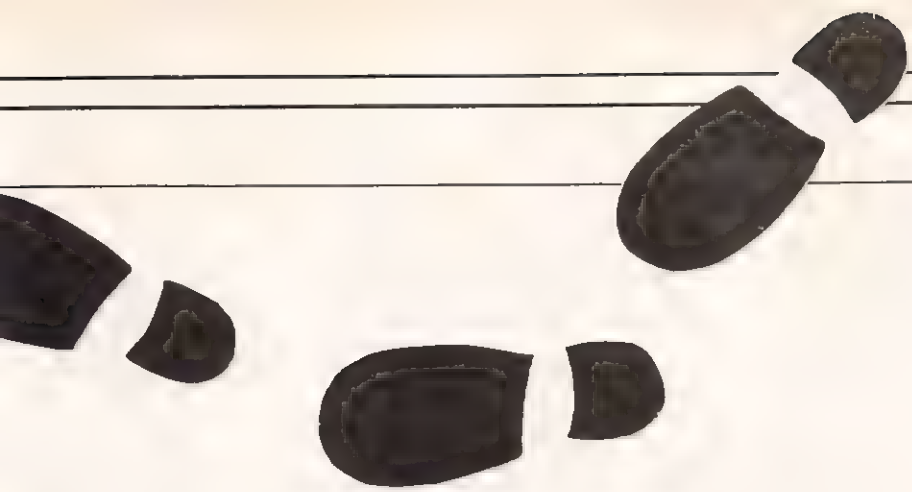
So inserting a REM at the start of a line results in all the commands in that line being ignored. You could achieve the same effect by deleting the line completely, but by using REM you still have the line intact on screen.

Returning things to the original state is simply a matter of deleting the REM rather than having to type in the line again.

You can, in fact, insert the REM in any part of the line after a colon. You can even bypass a whole subroutine by REMing the line which contains the GOSUB. This is a useful way of avoiding screens of instructions at the start of a game.

But be careful that no variables are set up in the routine, or you'll get some annoying results.

Another problem can occur when you've finished typing in a program. You'll sometimes get strange results on the screen but because things happen so fast it's often difficult to establish exactly where in a program a certain action is taking place. For



example, in one version of Digger I recently looked at there were all sorts of weird shapes dotted about the screen and none resembled the shapes that should have been there.

In order to solve this apparent disaster it was necessary to pinpoint when these "thingies" were being put on screen.

Finding this exact moment can be a bit hit and miss, even using the STOP hint we've discussed. The following routine should narrow down the areas of trouble by putting control of the whole screen display literally under your finger tips a frame at a time.

It is very important that you follow these steps exactly or you're going to have even weirder results than the ones you had in the first place.

Make sure your program is numbered in 10s, even if you have to renumber it yourself. You achieve this by typing in the command RENUM [Enter]. Now type in the following in direct mode, in other words without a line number:

```
KEY 139,"WHILE VAL(INKEY$) <> 1
:WEND" + CHR$(13)
```

This line sets up the small Enter key in the numeric key pad cluster with the instruction between the quotation marks. This instruction is used to make the program wait for the number 1 key to be pressed before it continues – the number 1 key next to Escape, not next to Shift.

What the instruction is actually saying is "While the value of the key being pressed does not equal 1 – do nothing" followed by an Enter – CHR$(13).

Next type in direct:

```
AUTO 5
```

"It is important to follow these steps exactly or you'll have even weirder results than those you had in the first place"

and then keep pressing the small Enter key. You will see the WHILE . . . WEND line being automatically entered in every alternate line ending in the number 5.

Continue until you reach a line number higher than the final line number of your original program. At this point in the proceedings if you type in LIST [Enter] you will actually see every alternate line containing the new instruction.

If you now type in RUN [Enter] you can single step through the program at your own speed and watch for something on the screen that you want to trap. Pressing Escape twice will jump out of the program at a line number very close to your mistake. It can't be the exact line number of course because you will always stop at a line ending in a 5.

While this routine may not pinpoint the error exactly, at least it will stop you in the subroutine that is causing the problem.

When you've finished, removing the odd line numbers is a simple matter. Set up the same Enter key to put in a space and an Enter as follows:

```
KEY 139," " + CHR$(13)
```

Now keep pressing Enter again and you will see all the line numbers ending in 5 scrolling up the screen with an asterisk * against them. The asterisk is not actually entered in the line, it is merely a symbol that that line did exist. Continue until you get the message "Line does not exist". This tells you that you have just deleted the last 5 line and you can carry on normally.

Well I think you've had enough practical tips to be going on with, but I'll end with a word of warning.

Armed with your newfound knowledge, there will be a great temptation to burn the midnight oil hammering away trying to find that one last bug.

This is great if things work out well but not so great if you're having a rough time. The programmer's bible states: "In every program there is always one more bug".

You want to hear our editor after a late night de-bugging session. "Got that damn program working at last lads. Took me till 3 this morning, but I finally cracked it", he says/mumbles/groans/grunts (delete word not required) propping eyelids open with two match sticks.

"Super", thinks I, "but we know we're going to pay for it later in the day when he's shattered".

Don't do it unless you are really under pressure. Give yourself a break. Make a cuppa, throw stones at next door's dog or get another program out and tinker with that for a bit.

But don't be too insistent on finishing it all in one session. I've seen, among programmers, more head banging than you'll see at a heavy metal concert and there really is no need for it.

Programming is fun, and the results can be very rewarding. Let's try to keep it that way.

See you next month.

"One way to find an error without destroying any part of your program is to REM the line you think is causing the trouble"

You wanna play
Then get in the

NIGEL PETERS examines that awkward channel parameter in Part VI of his series on CPC464 sounds

FIRST the good news. This month we're going back to our good old basic SOUND command.

Made up of:

```
SOUND channel,pitch,duration,volume
```

it ignores the envelopes and noise, so things should be a lot simpler.

The bad news is that the channel parameter isn't quite the simple beastie that I told you it was.

However, for the time being, let's stick to the old formula where a channel parameter of 1 produces a noise on channel A, while 2 uses channel B, and 4 plays on channel C.

Let's play a note on channel A with:

```
SOUND 1,200,100,7
```

This produces a note of pitch 200 which lasts for one second and is played at full volume. To play the same note on channel B we'd use:

```
SOUND 2,200,100,7
```

and on channel C:

```
SOUND 4,200,100,7
```

Exciting stuff isn't it? Using these three commands you may think that:

```
SOUND 1,200,100,7:SOUND2,200,
100,7:SOUND 4,200,100,7
```

would produce the same note at the same time on each of the channels. You'd be right.

But what does:

```
SOUND 7,200,100,7
```

produce? Try it and see. If you can find any difference between that noise and the one produced by the previous set of three sound commands you've got better ears than I have. (Which probably explains a lot about this series!)

The point is that:

```
SOUND 1,200,100,7:SOUND2,200,
100,7:SOUND 4,200,100,7
```

and:

```
SOUND 7,200,100,7
```

produce the same noise. The question is why? It must be something to do with the 7 in the channel parameter of the single SOUND command, but what?

After all, there are only three channels and we've got numeric labels for each of them. So what's happening? How do we get three notes playing at once from a single SOUND command?

Well, take a look at the channel parameter of:

```
SOUND 7,200,100,7
```

again. Notice that the sound is played on channels A, B and C. Now we know that the number labels for these channels are 1, 2 and 4. And guess what 1+2+4 adds up to?

Yes, it's the 7 that we had in the channel parameter of the solitary but very productive SOUND command.

From this you should be able to see that when we want a SOUND command to play a note on more than one channel, we add together the individual channel numbers to form a new, combined number.

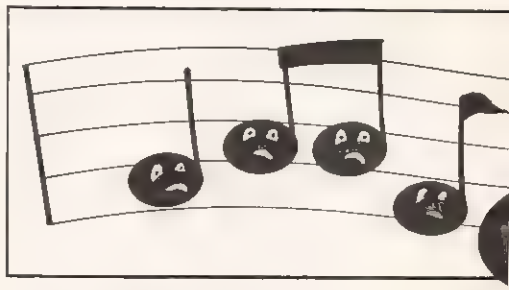
When the Amstrad finds this number it realises that you want the note to play on more than one channel and takes the appropriate musical action.

In the case of:

```
SOUND 7,200,100,7
```

the micro came across 7 in the channel parameter and immediately realised that this wasn't the normal 1, 2 or 4 it might have expected.

Being a computer, and therefore good at maths, it then broke the number down into the component 1=2+4 and played the note on



channels A, B and C.

So, by adding together the channel parameters we can make one SOUND command to do the work of two or three.

To play a note on channels A and C at the same time, we'd use a channel parameter of 5 (1+4) as in:

```
SOUND 5,300,50,6
```

while to play the same note on channels A and B we'd use a parameter of 3 (1+2).

Table I shows how the differing channel parameters bring different combinations of channels into play.

Now let's leave combined channel parameters and get back to our simple SOUND commands.

Try entering:

```
SOUND 1,200,100,7:
SOUND 1,300,100,7
```

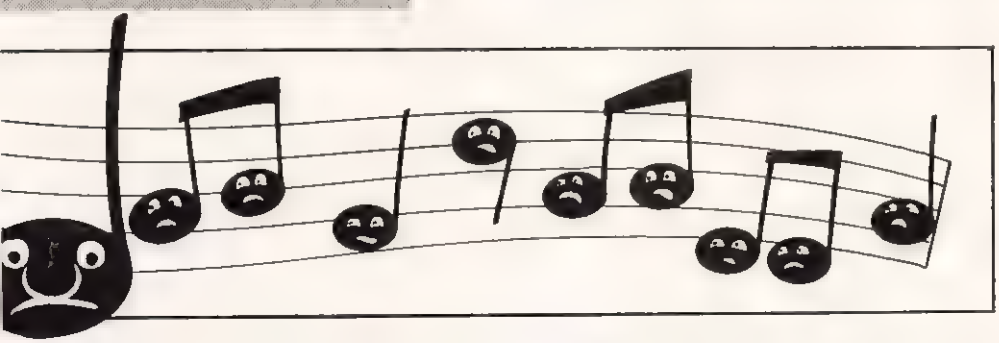
channel parameter	channel		
	A	B	C
1	X	/	/
2	/	X	/
3	X	X	/
4	/	/	X
5	X	/	X
6	/	X	X
7	X	X	X

X channel used
/ channel silent

Table I: Channel parameter combinations

a note?

queue



and see, or rather, hear what happens. You get a one-second note at pitch 200, followed by a one second note at pitch 300. This is more or less what you'd expect.

The Amstrad comes to the first SOUND command, processes it and starts playing the note.

While this note is playing, the micro comes to the second SOUND command and processes that. Now it has a problem.

The second SOUND command has told it to play a note on channel A but, in the very fast world of microelectronics, the first note is still playing.

What can it do? Should it interrupt the first note? Or should everything grind to a halt until the first note is finished and the second SOUND can be processed?

The answer is that the Amstrad lets the first note carry on playing for its full duration and pops the second on to what is known as a queue.

When the first note is finished, the micro looks at the queue, finds the second note stored there, and plays it.

The good thing about the queue is that the program can carry on doing other things, rather than waiting for SOUNDs to be processed.

If you don't follow that, try:

```
SOUND 4,1000,500,5:
SOUND 4,100,500,5:
PRINT "The Amstrad gets on with other
things"
```

Here you might expect from the order of the lines that you'll hear the first sound, then the second, then see the message.

In fact what happens is that the micro starts the first sound, puts the second on the queue for channel C and then goes on to the next statement.

This means that the first note is heard, the message appears and only when the duration of the first note is finished do we hear the second note.

The queue is used for storing notes while the micro gets on with processing any Basic statements that follow.

There is a separate sound queue for each channel, each totally independent of each other.

These queues, however, are limited in length. There is only room for one note to be playing and four notes to be stored on the queue. After that you've got problems, as shown by Program I.

As you can see, the FOR . . . NEXT loop is trying to play ten notes, one after another. After each note a message is to be printed.

It seems fairly straightforward, but if you expected a note to be played, then a message, then another note, then another message, you'll have been disappointed.

The first time round the loop the Amstrad plays the note and prints the message.

However, the next four times round the loop the first note hasn't finished playing.

What happens is that the four notes produced by these cycles are popped on to the queue waiting for their chance.

There are no problems with the PRINT commands, however. There's nothing to stop these appearing as normal, so we get the first five messages appearing before the first note has finished!

Now the Amstrad has problems. There's a note sounding, and there are four notes on the queue, waiting to play.

The queue is full, there's no more room, yet on the sixth time round the loop, line 30 is telling it to make a noise! Something has to give.

What happens is that the program grinds to a temporary halt waiting for a space to appear in the queue.

When the first note stops sounding, the second comes off the queue and is played. The remaining three notes "shuffle" along the queue leaving a space at the end.

Now the Amstrad can carry on, putting the sixth note on the end of the queue and printing the appropriate message.

The trouble is that the next time round the loop the micro comes up against the same problem, a SOUND to be processed and no room on the queue.

Again the program grinds to a halt until a space is free.

You can see that this stopping and starting explains why the first five messages appear in a flash while the next five come at one-second intervals as the notes change.

As I said before, there is a queue for each channel, each queue holding four notes as well as the one currently playing on that channel.

Each channel queue operates independently.

Having said that, though, once you attempt to put a note on a channel queue that's already full, then everything comes to a halt until a space appears for it.

In this way one channel's problems can mess up another channel.

Program II shows how a note can sound on one channel even though

```
10 REM Program I
20 FOR note= 1 TO 10
30 SOUND 1,100*note,100,7
40 PRINT "Note number "note
50 NEXT note
60 PRINT "The program is finished..."
70 PRINT "the music lingers on."
```

```
10 REM PROGRAM II
20 SOUND 1,100,5000,7
30 FOR note=1 TO 10
40 SOUND 2,100*note,100,7
50 PRINT "Note number "note
60 NEXT note
```

another has a full queue. Can you explain why?

Leaving queues for a moment, listen to Program III.

```
10 REM PROGRAM III
20 SOUND 1,239,100,5
30 SOUND 1,190,100,5
40 SOUND 1,213,100,5
50 SOUND 1,239,100,5
```

This consists of a four-note "tune". Each note is played on channel A one after the other, each note lasting for one second. The whole tune takes four seconds. Now bend your ear to the charms of Program IV.

```
10 REM PROGRAM IV
20 SOUND 2,119,100,7
30 SOUND 2,127,100,7
40 SOUND 2,159,200,7
```

This is made up of three notes played one after the other on channel B. The first two notes each last for one second, the last one for two seconds. Again, the whole tune lasts for four seconds.

Now, in the interests of harmony and to excite music-lovers everywhere, let's combine the two tunes. One way of doing this is shown in Program V:

```
10 REM PROGRAM V
20 SOUND 2,119,100,7
30 SOUND 1,239,100,5
40 SOUND 2,127,100,7
50 SOUND 1,190,100,5
60 SOUND 2,159,200,7
70 SOUND 1,213,100,5
80 SOUND 1,239,100,5
```

The way that the SOUND commands are arranged reflects the structure of the combined tunes. Line 20 starts a one-second note on channel B while line 30 starts another one-second note playing on channel A.

Since they are both on different channels, they are played simultaneously, and so a pleasing harmony is heard. (Well, I think it's pleasing.)

The next two lines again put a note on each channel, with line 60 putting a final two-second note on channel B, matched by lines 70 and 80 putting two one-second notes on channel A.

The whole tune lasts four seconds and, as we've kept below six notes on each channel, we've no queue problems.

Program VI plays exactly the same melody even though the SOUND commands are in a different order.

```
10 REM PROGRAM VI
20 SOUND 1,239,100,5
30 SOUND 1,190,100,5
40 SOUND 1,213,100,5
50 SOUND 1,239,100,5
60 SOUND 2,119,100,7
70 SOUND 2,127,100,7
80 SOUND 2,159,200,7
```

Here all the channel A sounds come before the channel B sounds in the listing.

The micro works its way through first four notes, playing one and putting the other three on the channel A queue.

Then it comes to the three SOUND commands for channel B. It processes these, playing the first and popping the other on to the queue for channel B.

This all happens so quickly that we hear the notes on channels A and B start at the same time, even though the statements that produce the notes are four lines apart.

To be really accurate, the Amstrad does take time processing lines 30, 40 and 50, putting these notes on the queue before it reaches the first channel B note and plays it.

This means that the notes don't start at exactly the same time. Having said that, I can't hear the difference and I doubt if anyone else can.

Previously we've had an example of how too many SOUNDs can mess up a Basic program, causing it to pause.

Program VII shows how the Amstrad getting on with Basic can mess up the pattern of SOUND commands that go together to make up tunes.

You'll see that apart from line 40 this is exactly the same as the previous program. Yet the tune is destroyed.

What's happened is that the micro has processed lines 20 and 30, playing one note and popping the next on the channel A queue.

Then it hits the delay loop and hurtles round 40 times doing nothing but taking up time.

Once the loop is finished, the program goes on to lines 50 and 60, popping these notes on the queue.

Finally the micro reaches line 70 and starts playing the first note on channel B, putting the next two on the channel B queue.

The problem arises because the first note on channel A has been playing for some time before the first note on channel B is started.

Previously they began at the same time (or as near as made no difference). Now, however, the delay loop has slowed down the program so much that by the time the micro reaches line 70 everything is out of step.

Try changing the length of the delay loop to see the effect.

You might say: "Well don't stick delay loops in your tunes and everything will be all right". It's a fair point, but remember we can only have five notes at a time on a channel without causing problems.

However, any decent tune has a lot more than five notes so if we want musical accompaniment to our programs we have to grab five notes, do something else, then grab other notes as space appears in the queues.

The delay loop in Program VII symbolises these "something else"s.

As you can see, while the program is doing these other things, tunes can get out of step. Don't worry, though, we're not stuck to a repertoire of five-note ditties.

Program VIII shows how to over come the problem.

As you can hear, the tune is intact,

```
10 REM PROGRAM VII
20 SOUND 1,239,100,5
30 SOUND 1,190,100,5
40 FOR delay=1 TO 200:NEXT delay
50 SOUND 1,213,100,5
60 SOUND 1,239,100,5
70 SOUND 2,119,100,7
80 SOUND 2,127,100,7
90 SOUND 2,159,200,7
```

```
10 REM PROGRAM VIII
20 SOUND 17,239,100,5
30 SOUND 17,190,100,5
40 FOR delay=1 TO 200:NEXT delay
50 SOUND 17,213,100,5
60 SOUND 1,239,100,5
70 SOUND 18,119,100,7
80 SOUND 18,127,100,7
90 SOUND 18,159,200,7
```

It takes two to make a rendezvous, and both have to recognise each other

despite the delay loop. The secret lies in the funny-looking channel parameters.

Remember how we can add together our channel numbers to form new, more powerful channel parameters?

Well we can use the same sort of technique to ensure that notes in different channels are playing in step. We just add 8, 16 or 32 to the appropriate channel parameter as needed.

Suppose we wanted a note on channel A to be played starting at exactly the same time as a note of a different pitch on channel B.

What we do is to cause the notes to "rendezvous" with each other by adding the relevant number to the channel parameter.

To get a note to wait until a note is ready on channel B we add 16 to its channel parameter.

We want a note on channel A to coincide with a note on channel B, so we add 16 and 1 to get a new parameter 17.

So:

```
SOUND 17,100,100,5
```

produces a note on channel A that will only play when there's a note on channel B.

That seems simple enough, so, if you haven't already, enter:

```
SOUND 17,100,100,5
```

and listen to what happens.

Nothing should happen, as adding 8 to the parameter has told the micro to wait for a note on channel B and there isn't one, yet. So let's give it a note on channel B with:

```
SOUND 2,4000,100,5
```

You'll probably be disappointed that all you hear is the low note on channel B. The high note waiting on channel A hasn't put in an appearance, even though we've given it a note on channel B. Has something gone wrong?

The answer is that it's our fault. We've added 16 to the first note's channel parameter to tell it to wait for a note on channel B but we haven't marked out *which* note on channel B.

It takes two to make a rendezvous and both have to recognise each other.

Not only must the first note have a number added to its channel parameter, the second one has to as well.

It's rather like two people meeting for the first time. One will have a bowler hat on, the other will be carrying a copy of The Times.

In the example above, the note on channel A was wearing its bowler hat (17) but the note on channel B wasn't carrying its copy of The Times. As a result, they passed each other by.

What we should have done was add 8 to the channel B note's parameter. If my maths is correct 2+8 gives 10 so:

```
SOUND 10,4000,100,5
```

should not only produce a low note but also bring the high note on channel A out of hiding.

Try it and see.

To sum up, two notes can be made to start at the same time or rendezvous by adding the appropriate values to their channel parameters.

To get a note to wait for one on channel A before we start, we add 8.

For a rendezvous with channel B we use 16 while a date with channel C has us adding 32.

Table II summarises the situation.

Addition	Rendezvous with
8	Channel A
16	Channel B
32	Channel C

Table II: Rendezvous factors

Using this, we can have a note on channel A coinciding with one on channel C using:

```
SOUND 33,600,100,7
```

and:

```
SOUND 12,3500,100,7
```

Similarly, to link notes on B and C we use:

```
SOUND 34,12,100,7
```

with:

```
SOUND 20,3750,100,7
```

Using your new-found knowledge, you should now be able to see how the 17's and 10's of Program VIII work to overcome the delay loop.

The note on channel A can't start playing until it comes across another marked for rendezvous on channel B. This means that the notes stay in step.

And that's it for this month. We'll be exploring the channel parameter in more detail next time.

Until then, play around with the values given in Tables I and II and seen how they work in practice.

You may find that it helps to have set up the small Enter key with:

```
KEY 139,"SOUND 135,0,0,0"+CHR$(13)
```

Now when the sound channels get out of hand just press the small Enter and all the garbage will be cleared. I'll explain why in a future issue.

And, when you're more familiar with rendezvous, perhaps you'll be able to see why Program VIII suffers a bit from "overkill". Have fun.

Make sure you don't go out of bounds

Third and final part of KEVIN EDWARDS' series on Amstrad Basic animation

IN this, the last of the present series on animation, we'll go through a few more interesting methods which help improve animation.

As you know, the Amstrad has two cursors, one for text the other for graphics. Normally the PRINT command outputs text to the text cursor (the big square blob). But by using the command TAG you can force all the PRINTs to output text to the graphics cursor instead. TAGOFF reverses the effect.

The graphics cursor, which is invisible, can be positioned at any pixel point on the screen, unlike the text cursor which can only be moved to defined character cells.

This means that once TAG has been executed the text can also be printed from any pixel point. In addition, you can select the way in which the text is placed on the screen, the three different ways being EOR, AND and OR.

These logical operators, as they are known, are used to determine the colour of the pixel to be displayed. For our purposes we'll only be interested in EOR.

Let's consider the case where the text is being printed at the graphics cursor, EOR is being used to calculate the resultant colours and we're in Mode 1.

If we have a blue background (colour 0) and we print a yellow character (colour 1) the resultant character will be yellow because 0 EOR 1=1 (yellow).

But if we now display the same character – with the same colour – over this one the previous character will disappear, because 1 (yellow) EOR 1 (yellow) =0 (blue). So instead of displaying another character on the screen we erase the previous one.

Using the EOR option allows us to display a character on the screen and then erase it, by repeating the same operation. It's our old familiar now we see it, now we don't, using a new technique.

Program I does exactly this. Use the Z and X keys to move Smiley left and right.

Line 30 selects the EOR printing mode.

Line 40 initialises the X coordinate to 0.

Line 50 displays Smiley on the screen using a subroutine formed by lines 90-130.

Line 60 tests to see if the Z key is pressed. If it is, the character is deleted (the first GOSUB 90) and moved left by one character cell (X=X–16). Now Smiley is displayed in his new postion (the second GOSUB 90).

These steps are not executed if Smiley is at the left edge of the screen, when X=0.

```
10 REM PROGRAM I
20 MODE 1
30 PRINT CHR$(23);CHR$(1)
40 X=0
50 GOSUB 90
60 IF INKEY(71)=0 AND X>0 THEN GOSUB
90:X=X-16:GOSUB 90
70 IF INKEY(63)=0 AND X<623 THEN GOSU
B 90:X=X+16:GOSUB 90
80 GOTO 60
90 MOVE X,159
100 TAG
110 PRINT CHR$(224);
120 TAGOFF
130 RETURN
```

Program I

Line 70 tests the X key. Again, GOSUB 90 is used to delete and display Smiley. The character is moved right by one character cell by adding 16 to X. This time the character is not moved if it's at the right hand edge of the screen.

Line 80 'jumps' back to line 60.

Line 90 MOVEs the graphics cursor to Smiley's current position.

Line 100 makes all text go to the graphics cursor.

Line 110 prints a Smiley character at the graphics cursor.

Line 120 restores text output to the text cursor.

Lines 90-130 are used to display and delete the character. This saves

repeating similar program lines.

To show you one of the disadvantages of EOR add the following line to Program I:

```
25 LOCATE 20,16:PEN 3:PRINT "EOR IS
CLEVER"
```

Now see what happens when you move Smiley over the text. Cyan appears (colour 2) where Smiley and the text overlap.

But where did the cyan come from? Nothing was printed in cyan! In fact, cyan appeared because 1 (yellow Smiley) EOR 3 (red text) = 2 (cyan).

If this seems double Dutch to you don't worry, the colour changes are made by the CPC464. You can learn more about EOR in May's Bits and Bytes – but you don't have to understand it to use it!

To change the subject, any moving object has its movement restricted by something. On the Amstrad, the restrictions are the edges of the screen display.

The easiest way to stop a moving object going "out of bounds" is to keep a record of its position. Knowing where an object is allows you to check that any future movement doesn't move outside the defined limits.

These limits don't have to be the edge of the screen. They can be any part of the screen you choose.

Program II bounces a ball diagonally around the Mode 1 screen. The variables X and Y contain the x and y coordinates of the ball. These correspond to the text coordinate positions on the screen.

Let's take a look at how it works.

```
10 REM PROGRAM II
20 MODE 1
30 X=5:Y=5
40 ON INT(RND(1)*4)+1 GOTO 50,60,70,8
0
50 RX=-1:RY=-1:GOTO 90
60 RX=-1:RY=1:GOTO 90
70 RX=1:RY=-1:GOTO 90
80 RX=1:RY=1
90 IF (X+RX)<1 OR (X+RX)>39 THEN 40
100 IF (Y+RY)<1 OR (Y+RY)>24 THEN 40
110 LOCATE X,Y:PRINT CHR$(32);
120 X=X+RX:Y=Y+RY
130 LOCATE X,Y:PRINT CHR$(231);
150 GOTO 90
```

Program II

Line 30 initialises X and Y.

Line 40 selects a random number between 1 and 4 and jumps to line 50 if the number is 1, 60 if the number is 2 and so on. This selects a random direction for the ball.

Lines 50 to 80 define the relative change in direction of the ball for four different directions. RX is the relative change of the X coordinate, RY is the relative change of the Y coordinate.

For example, line 70 indicates that the change in X will be 1 (RX=1 means move the ball right by one character cell) and the change in Y will be −1 (RY=−1 means move the ball up one character row).

Moving an object right and up is the same as moving the ball north east where north is the top of the screen.

Line 90 checks to see if the X coordinate will still be on the screen if the ball is moved in the new direction. If the ball would move off the screen a new direction is selected – the program jumps to line 40.

Line 100 does the same as line 90. This time it checks the Y coordinate.

Line 110 positions the text cursor at the ball's previous position and deletes it by PRINTing CHR$(32) – a space.

Line 120 calculates the ball's new position.

Line 130 displays the ball.

Line 150 continues the movement in the same direction by jumping to line 90.

At the moment the ball will only change direction if it hits the edge of the screen. If you add the following line the ball will also have the opportunity of changing direction whenever it wants. See if you can work out why!

```
140 IF INT(RND(1)*20)=10 THEN 40
```

Program III checks for boundary collisions. This time the ball is moved to the opposite side of the screen when a boundary is met instead of changing direction.

Lines 100 to 130 are responsible for checking if the ball is out of bounds. If it is, the X and Y coordinates are adjusted accordingly.

For example, if a ball is at X coordinate 40 – the right hand edge of the screen – and it moves north east, the new X coordinate would be 41.

But this is off the screen – Mode 1 has only 40 characters per row, so line 110 would reset X to 1 – the left edge of the screen. The same happens if the Y coordinate goes off the screen.

If you think you're clever enough you can try and make the ball move vertically and horizontally – not at the same time though.

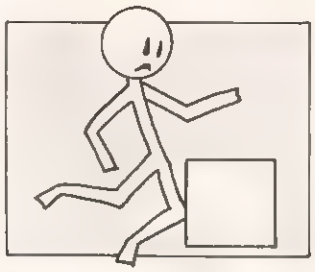
Finally we come to Program IV. Type it in and RUN it. All you'll see is Smiley walk across the screen until ▶

```
10 REM PROGRAM III
20 MODE 1
30 X=5:Y=5
40 LOCATE X,Y:PRINT CHR$(231)
50 ON INT(RND(1)*4)+1 GOTO 60,70,80,9
0
60 RX=-1:RY=-1:GOTO 100
70 RX=-1:RY=1:GOTO 100
80 RX=1:RY=-1:GOTO 100
90 RX=1:RY=1
100 IF (X+RX)<1 THEN GOSUB 190:X=40:G
OTO 160
110 IF (X+RX)>39 THEN GOSUB 190:X=1:G
OTO 160
120 IF (Y+RY)<1 THEN GOSUB 190:Y=24:G
OTO 160
130 IF (Y+RY)>24 THEN GOSUB 190:Y=1:G
OTO 160
140 GOSUB 190
150 X=X+RX:Y=Y+RY
160 LOCATE X,Y:PRINT CHR$(231);
170 IF INT(RND(1)*15)+1=10 THEN 50
180 GOTO 100
190 FOR delay=1 TO 50:NEXT delay
200 LOCATE X,Y:PRINT CHR$(32);
210 RETURN
```

Program III

```
10 REM PROGRAM IV
20 MODE 1
30 X=1:Y=14
40 LOCATE INT(RND(1)*20)+15,Y
50 PRINT CHR$(143)
60 LOCATE X,Y
70 PRINT CHR$(224)
80 IF TEST(X*16,((26-Y)*16)-1)<>0 T
HEN END
90 FOR delay=1 TO 50:NEXT delay
100 LOCATE X,Y
110 PRINT CHR$(32)
120 X=X+1
130 GOTO 60
```

Program IV



he hits a randomly positioned block.

What's so clever about that? The answer is the collision detection.

What you have to do is test if the character can be moved to the next character cell. This is done by looking at the top left corner of the character cell to see if any pixels are set. We do this with the appropriately named TEST command.

TEST x,y returns the colour of the pixel at graphics coordinate x,y. If the colour of the pixel is the same as the background then it is safe to move the character, for nothing has been hit. Otherwise, a foreground colour had been detected, indicating that an object has been found.

The problem with this is that the graphics x and y coordinates are not the same as text coordinates. Take a look at Diagram I.

So what we must do is convert a text coordinate into a graphics one so that we can test its pixel colour.

The following statement will return the colour of the top left pixel of a character at text coordinate X,Y – Mode 1 only:

```
pixcol=TEST((X-1)*16,(((26-Y)*16)-1))
```

Note that only the top left pixel of the character is tested. If you look at line 80 you'll see the equation.

If you're wondering why the (X–1)*16 is X*16 in the program, remember we're checking one character ahead in the X axis direction. So (X–1+1)*16 becomes X*16.

Line 40 randomly positions a block on the same line as the stick man.

You should be able to understand the rest of the program by now.

And that's the end of this series. I hope it has given you more ideas for your own animation.

As you've seen, animation is quite easy and can be achieved by using very simple Basic commands. Now it's your turn to write your own animated programs. I look forward to seeing some of them.

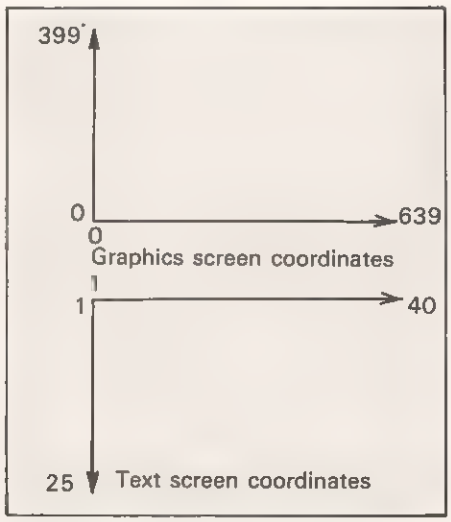


Diagram I: Screen coordinates for Mode 1

Making a fine program better

AS a new reader, the article which took my eye in the February issue of Computing with the Amstrad was the Word Processor by Roland Waddilove called Text Editor.

Having just invested in an Amstrad CPC464, this program seemed to be just what I was looking for without having to make a capital investment to one of the software houses.

Undaunted by my previous experiences of hours of typing in programs, only to find that they either didn't run at all or were full of errors, I decided to type in the program.

You can imagine my amazement when it ran first time (after correcting my own typing errors) and in fact did everything that the article said it would.

After using the program for some time to type some letters, I realised that it could be improved considerably by the addition of a couple of extra options:

1. An audible beep near the end of each line.

2. The ability to print out selected lines of text.

Thanks to Mr Waddilove's use of descriptive variables, I was able to accomplish all of these.

To provide an audible beep near the end of each line, the solution was quite simple. The addition of the following line to the program is all that is required:

```
475 IF char$<>"" AND x%>len
gth-10 THEN PRINT CHR$(7)
```

This line sounds a beep as soon as you are within ten characters of the end of line.

To enable selected lines to be sent to the printer, add the new lines numbered 2301-2309 (Listing 1) and also line 2415 to the program.

This routine works by modifying the variable (here) to point to the start of text that we wish to print. This is achieved by lines 2301 to 2309.

Line 2415 restores the original value of (here) before returning to the main program. – **Peter Whiteley, Bramhall, Stockport.**

■ Thank you for the improvements. We're sure our readers will benefit from them.

```
2301 temp1=here:temp2=finis
h
2302 IF length<1 THEN lengt
h=1
2303 LOCATE 10,4:PRINT"Tota
l lines available for print
ing are ";(finish-here)/len
gth
2304 LOCATE 10,8:INPUT"Ente
r line number to start prin
ting from.";line1%
2305 LOCATE 10,12:INPUT"Ent
er line number to finish pr
inting at.";line2%
2306 IF line2%<line1% THEN
LOCATE 10,16:PRINT"Line num
bers entered incorrectly.":
FOR de=1 TO 3000:NEXT:CLS:G
OTO 2300
2308 here=here+(length*(lin
e1%-1))
2309 finish=here+(length*(l
ine2%-line1%))

2415 here=temp1:finish=temp
2
```

Listing 1

On right lines

SINCE purchasing the Amstrad CPC464 last June, I have been rather disappointed, to say the least, with the magazines and literature in general regarding this excellent machine.

I would therefore like to take this opportunity to congratulate you on the publication of Computing With the Amstrad. It's a superb well-thought-out magazine which provides entertaining reading from start to finish.

Your articles such as "First Steps", "Machine Code" and "Bits and Bytes" have been described in such a manner that what once seemed a task not to be undertaken has turned into an enjoyable "course".

I should also like to commend you on the listings in the magazine and give Mr Waddilove a pat on the back. – **J.F. Watson, Barking, Essex.**

DIP's the key

CAN your A Team help me with a problem I have with the Brother M1009 printer? – double space line feed?

According to the "Amstrad CPC464 User" the lead to Pin 14 needs to be cut to stop these extra line feeds.

How is it done? Can I do it myself? Is there any risk of damage to the cable or printer? – **P. Andrews, Conisbrough, Doncaster.**

■ You should be able to switch off the automatic line feed by setting one of the DIP switches on the printer.

Accessing codes

HAVING just bought an Amstrad CPC464 computer, it having succeeded the Spectrum, I am writing to try to acquire some information regarding connecting the computer to my Tandy DMP 105 printer.

This letter has been produced by the printer, which I also use with a BBC Micro at the school where I am a teacher, and I am pleased with the way in which it handles word processing using the Easi-Amsword program.

I am, however, disappointed that the Amstrad will not allow CHR$ codes above 127 to be sent to the printer thus limiting its potential to letters only.

The DMP 105 printer supports not only a comprehensive graphics selection but also the European character set which is useful for producing foreign-language material.

I would be grateful if you could advise me on how I can access the CHR$ codes greater than 126 on the Amstrad, preferably from the software.

May I close by congratulating you on a most excellent publication. – **John R. Steel, Darlington.**

■ You are one of many who have a printer with inaccessible codes. Many of the useful functions available on printers are accessed by sending characters with codes above 127.

As of yet, we have found no way of printing such codes using software. This is because the problem lies within the Amstrad itself.

What happens is that bit 7 of each byte sent to the printer is ignored! Why this was done still remains a mystery to us all.

Perhaps a hardware modification of some kind will solve the problem.

Is there anyone out there who can assist?

Utilities for the DD1

THE following listing is for the Amstrad CPC464 computer. It is a utility program for use with the DD1 disc drive.

ERA=erase any program of

your choice.

REN=RENAME any program.

*DIR=directory listing of any chosen suffix groups, ie *.BAK will give all programs on disc with BAK suffix.*

LST=any chosen program from CAT menu.

After typing in the listing, SAVE "ERDL". To use, RUN "ERDL". The disc contents will then be displayed and the utility program will be ready to use. Just follow prompt instructions. – **Paul Reynolds, Sissinghurst, Kent.**

```
10 MODE 2:WINDOW #1,1,80,25
,25
15 BORDER 0:INK 0,26:INK 1,
9:PAPER 0:PEN 1
20 CAT
30 PRINT#1,"Enter COMMAND (
ERA, REN, DIR or LST): ";:L
INE INPUT#1,C$:CLS#1
40 LET C$=UPPER$(C$)
50 PRINT#1,"Enter FILE NAME
 to be "+C$+"'ED: ";:LINE I
NPUT#1,F$:CLS#1
60 IF C$="ERA" THEN |ERA,@F
$
70 IF C$="REN" THEN PRINT#1
,"Enter NEW FILE NAME for "
+F$+": ";:LINE INPUT#1,N$:C
LS#1:|REN,@N$,@F$
80 IF C$="DIR" THEN CLS#0:C
LS#1:|DIR,@F$:GOTO 30
90 IF C$="LST" THEN OPENIN
F$:WHILE EOF=0:INPUT#9,N$:P
RINT N$:WEND:PRINT:PRINT:PR
INT#1,"PRESS 'ENTER' KEY TO
 CONTINUE.....";:LINE INPUT
#1,C$
100 GOTO 10
```

Hints needed

I HAVE recently bought Mission One (Project Volcano), Message From Andromeda and Arnold Goes To Somewhere Else (Nemesis), all adventures, but unfortunately I have never played any adventure before.

In Andromeda I have collected all of the objects except the coin and the metal bar, but I can't get back into the ship (even with the airlock open).

I can't break the beam which is holding the ship. I have tried to use the detonator but it has no effect.

In Mission One I can't get past the grill in the ventilation shaft. I have unscrewed it but I can't go forward as I will get electrocuted. Can anyone help me?

Finally, I can't get past the lake side in Arnold Goes To Somewhere Else. I have tried to use the carving and the paddles but they have no effect.

I would be grateful if anyone can get me out of this predicament. – **P. Smith, Whitefield, Manchester.**

Eye on the logic

FUNNY how much AI knows about me. I am all the people he describes in his excellent debugging article.

I have a little tip that may prove useful in this context. When debugging even a short line such as:

```
1240 dat=PEEK(&9FFC+(ASC(CH
R$(n))-32)*8+tat)
```

it is all too easy to lose the original logic. I make a copy at line 1241 and REM 1240 out with a shifted 7 (" ' "). Once de-bug has gone, usually a bracket too many or too few, renumber to 1240 and delete 1241.

One final point. Thank you, Mike Bibby. All previous machine code tutors have, after the first couple of pages, accelerated away and left me behind. To Mr Drury, who says Mike is too slow, Grrrr . . . – **L.G. Barclay, Acton, London.**

Alien impasse

WILL somebody please help us with Gems of Stradus? Having got totally besotted with the game, we have now come to a halt. We have encountered the Alien and can go no further.

Have we missed a room on the way which holds the answer? We feel not. So is there anyone who can assist us before we all lose our sanity? – **T.G. Stephens, Sittingbourne, Kent.**

Clogged up by garbage

I WONDER if you could throw some light on a problem? I typed in the game "Mad Adder" from the April issue and it requires the use of the Escape key to exit from the game.

However, I found the procedure affected the cassette handling routines whereby the LOAD", RUN" or CAT connect took some 15 seconds to respond with the prompt.

Interestingly if I replied with the Escape to the prompt, the delay became 30 seconds, but thereafter remained at 30 seconds.

By using STOP at various points in the program I narrowed the problem to the A$=CHR$(233) in line 49 and line 73. The problem remained until I reset the machine, loaded MAD ADDER (without running it) or loaded and/or run another program.

I wrote a simple program, listing attached, and found after running this the delay to LOAD" etc was about 1 minute 20 seconds to 1 minute 25 seconds.

I started experimenting by removing certain instructions and found the removal of SP$(X,Y)=A$ cured the problem. I noticed that line 73 was called 187 times in Mad Adder and line 60 840 times in my program and I wonder if this could have caused the different delay times.

I further experimented and found removing line 30 solved the problem. I changed line 30 to 'a$=CHR$(65)' and the problem appeared again.

However re-loading, changing line 30 to 'a$="A", in effect the same as before and running did not produce the problem. I applied these many changes to Mad Adder with the same effect. Is there anything wrong with my machine? – **J. Barrett, Reading, Berks.**

```
10 CLS
20 DIM sp$(40,21)
30 a$=CHR$(233)
40 FOR x=1 TO 40
50 FOR y=1 TO 21
60 sp$(x,y)=a$:LOCATE x,y:P
EN 1:PRINT a$
70 NEXT
80 NEXT
```

■ This is quite an interesting problem which often occurs when loading or saving programs, and is caused by garbage.

Garbage is redundant copies of variables – mainly strings, caused by extending a string beyond its initial length for example.

These must be removed for operations like loading and saving which require a 2k buffer to be set up for input and output to disc or tape.

To prevent this use:

```
10 OPENOUT "Dummy"
20 MEMORY HIMEM-1
30 CLOSEOUT
```

at the start of your program which will set up a permanent buffer which is ready whenever it is required.

No Escape . . .

PLEASE can you tell me if there is any way to disable the Escape key on the Amstrad CPC464. I know and have used the function of the ON BREAK GOSUB. It does not disable the Escape key on INPUTS. Please can you tell me if there is any way to disable the Escape key on INPUTS. – **M. Rogers.**

■ As you say, ON BREAK GOSUB does not disable the Escape key on input. The only way round this is to use

INKEY$ instead for any input. Does anyone know how to completely disable the Escape key?

Right on, Roland!

I MUST congratulate you on an excellent magazine. I've had my CPC464 since last November after progressing through from ZX81 through Texas T199/4a and Spectrum. My main reason for writing is to congratulate Roland Waddilove on his excellent clear and easy-to-follow Star Fleet listing in the March issue.

I've found with most listings from other magazines that they don't make full use of the Amstrad's capabilities.

Yes, sir, Roland is on to a winner as far as I'm concerned. Keep up the good work. – **Dave Clarke, Streatham, London.**

Foreign accents

I AM a computing beginner. How can I allocate the foreign accents such as " (CPC464 character 26), ' (39), , (44), " (162), ß (177) etc. directly and permanently to the lower-case keys on the top row of my keyboard?

I would still have the numbers on the separate right-hand pad.

Also, once allocated can they be superimposed on to a, e, u, n, etc., to produce ä, è, ü, ñ etc?

Without this my translation-processing is impossible.

HELP!!! – **Tony Shaw, Bridgwater, Somerset.**

■ Some of the characters in the character set are not available from the keyboard, but they can be printed using CHR$(X).

Any key can be programmed to produce any of the characters using KEY DEF.

KEY DEF 69,1,66 will change the "A" key so that it produces "B".

So select the characters you want and program them to the keys required.

Computing with the AMSTRAD Postbag

WE welcome letters from readers – about your experiences using the CPC464, about tips you would like to pass on to other users . . . and about what you would like to see in future issues.

The address to write to is:

Postbag Editor
Computing with the Amstrad
Europa House
68 Chester Road
Hazel Grove
Stockport SK7 5NY

Bar graphs

I BOUGHT an Amstrad CPC464 in July of last year and am over the moon despite the latest offer of free software which was a bit of a choker for this lad.

I would like to answer a query put by S.G. Squires in your February issue about filling bar graphs. Although a solution was supplied by "Computing With the Amstrad", it was (unlike the mag itself) a bit naff. Here's my solution:

```
5 CLS
10 INPUT "A number";n
20 FOR i=1 TO 5
30 x=i*60
40 z=15
50 y=140
60 FOR p=0 TO n
70 PLOT x,y+p,z
80 DRAW x+44,y+p,z
90 PLOT x+44,y,z
100 NEXT p
110 NEXT i
```

*My program as it stands will prompt for a number that will be the value determining the height of the five bars drawn on the screen. The number of bars is controlled in line 20, and if you find they're crowded off the screen you can make the bars slimmer by adjusting x + "whatever" in lines 80/90, and the spacing by altering i * "whatever" in line 30.*

The base line (zero) is positioned up or down the screen by line 50. Lines 60-100 work the oracle by drawing the bars line by line up to the value "M" in line 60. Variable "Z" controls colour value; and for minus values just replace line 60 thus: "For P=M to 0". – **Tony Howe, Purley, Surrey.**

Special offers

HAVING purchased an Amstrad CPC464 with a colour monitor just before Christmas, I am most disconcerted about the recent advertising, as regards the free software.

My husband and I paid cash and were offered no discount, the children (4) had saved hard for many weeks to put their contribution towards it. Then this offer comes out and it is enough to make one really lose faith in Amstrad.

Apart from this grievance we are more than happy with the computer, but I will really think twice when Amstrad bring another product on to the market. I really think this was a mean trick to play on the people who were faithful to Amstrad and had the confidence to buy a "new" computer on the market. – **Mrs Laura Savage, Ware, Herts.**

■ Having been caught by this sort of things ourselves, we know how irritating it can be. However, without wishing to toady to Amstrad, we can also see their point of view.

When you launch a new computer, no matter how satisfied you are with its quality, you can never predict how much of the market it will capture. Consequently you have to price pessimistically. Once it's established, you can then lower prices, or give away extra goodies.

And, of course, a manufacturer has to respond to his competitors. If they slash prices, he has to do something.

Obviously, this is of little comfort to early buyers of a micro. However, it does mean a larger user base for the micro in the end – which can only benefit Amstrad owners.

And your Amstrad, even without the software, is still extremely good value for money.

Monitor choice

I WAS interested in the letter from Mr Quemby (April issue) on the subject of a suitable colour TV/RGB monitor for the Amstrad. I recently trod the same path myself.

My requirements were very similar to his. Having decided on the green-screen Amstrad to cater for my own main interests of word processing, handling of club records and adventuring in text only, I also purchased a portable TV/RGB monitor to serve as a second television for the family and also to satisfy my eight-year-old son, who requires colour when "zapping the aliens".

Having studied the market I narrowed the choice down to two – the Philips 2007 and the Ferguson TX – both 14in colour portables and both, I was assured, RGB-compatible with the Amstrad.

I purchased the Philips for its TV picture quality (although

this is purely subjective) and have been more than satisfied with the resolution and colour rendition of the computer graphics.

A further "plus" – and I don't know whether or not this applies to all other TV/monitors – is that the Philips has both CVBS and RGB inputs and comes complete with a 5ft RGB connecting lead (6-pin DIN at each end). – **B. Johnson, Leeds.**

Sorcery's end

I SAW the advert for Sorcery in your April issue and bought the game. It's fantastic and very addictive.

After ten hours I finally completed it on Sunday March 17. Please tell me – am I the first to complete the Amstrad version?

My highest score is 82500. – **Richard Harrison, Doncaster.**

■ It is always deadly to claim you're the first at anything! You are, however the first we've heard of.

Judging by the scramble in our office for who's going to borrow our copy over the weekends, Sorcery certainly is something special!

CB networks

DO others find the same problem as myself when I try to use my computer in conjunction with CB networks for display and updating?

This is the heavy interference experienced which makes "copying" all but very local stations nigh impossible.

Any constructive suggestions as to what I can do with my rig or computer, which might ease the problem, will be much appreciated by myself and many of my friends around the country. – **Frank West, Ipswich, Suffolk.**

■ As I'm sure you're aware, you are suffering from stray signals which are being generated by your CPC464 (mainly from the monitor).

Three possible remedies which will reduce the interference are:

1. Earth your CB rig with a heavy-duty copper cable which is sunk approximately 1½ metres into the ground.

2. Screen your coax cable. RG8 coax cable will provide much better screening than more standard RG58.

3. Use a separate power source for your CB such as a car battery.

We hope this helps reduce those stray harmonics.

Putting a business on disc

I HAVE launched my own business venture this month, and to help me in this have purchased a CPC464, printer and disc drive. Brilliant micro isn't it?

Now, this is where Mr Bibby may be of assistance. The article written about the DDI-1 was kin to a bright light at the end of a long tunnel.

Being a clerk/typist and unemployed for nearly three years, I have taken my wife's advice and "gone it alone". Up to press it's been a tremendous hit.

One of my clients happens to be a builder. Now can you tell me how many individual, confidential reports I can keep on one disc.

Can I keep invoices, VAT accounts, staff records etc, etc, together? In other words, can I keep one small business to one disc?

Also, can you explain how modems can help in business. I've heard a bit about them and wondered if there is one available for the CPC464. – **Peter Andrews, Doncaster, South Yorks.**

■ Asking what you can fit on a disc is a bit like asking how long a piece of string is. To answer it properly, you'd have to have some knowledge of how disc files are constructed – not that they're difficult to understand. We'll be starting an easy-to-follow series on tape and disc files in a forthcoming issue.

A modem – several of which are now appearing for the Amstrad – would be useful for a business if you had salesmen, engineers, etc who could keep in touch with "home base" via a modem. And, of course, you could use the business information available on Prestel.

The risky road to translation

I WAS interested to see your Postbag request for an article on translating Spectrum Basic to Amstrad Basic. I hope that you and the potential contributor realise the risks you may be taking.

I've already tried to make the conversion but, in several cases, without success. Ask the CPC464 to perform some of the Sinclair functions and it stumbles, coughs and responds with "Syntax error" or some other unwanted epithet.

In exposing this, the risks are:

1. That Sir Clive will offer free C5s for life, for extolling the virtues of his micro.

2. That Amsoft will refer your action to headquarters and Amstrad Electronics will administer an official rebuke for daring to question their Basic.

It happened to me, although I haven't yet asked Sinclair for my trike.

My first problem, which provoked the negative reaction from Amstrad, was to find an Amstrad equivalent of the Spectrum's VAL. This versatile function on the Spectrum will evaluate a mathematical formula and is used in this way in many published programs.

For example:

```
10 LET x=PI/6
20 LET a$ = "x^2 + x*sin(x)"
30 PRINT VAL(a$)
```

on the Spectrum produces a value of 0.535907, which is the result of the formula in line 20, with the value in line 10 assigned to it.

On the CPC464, the answer is 0, since the Amstrad VAL produces only the leading numeric group in a string. If the leading characters are not numeric, as above, the answer is zero.

INPUT is another problem area in translation. Try to INPUT a mathematical string, of mixed alpha and numeric characters, as in line 20 above, and Amstrad Basic requires it to be defined as a string variable.

It is then unusable in mathematical operations on the variable. Just another example of Spectrum Basic's versatility!

*Another example? Say to the Spectrum GOSUB (n+10*y) and it does so quite happily. This makes for more efficient programming in some circumstances and is used in many Spectrum listings.*

Amstrad Basic recognises only numeric GOSUBs. Variables are coughed up – invariably!

Although encountered more rarely in programs, the Spectrum function TAN is not subject to the same numerical limitations as on the Amstrad.

Ask the CPC464 to plot a tangent curve, over a wide range of angles, and it will stop with an error report. The Spectrum carries on plotting until it returns to values within the screen limits.

Spectrum to Amstrad translation is not so fearsome as it appears, otherwise. A small Basic extension ROM for the Amstrad would cure everything!

I've no desire to knock the CPC464. It's still extemely good value for money and, in most respects, outshines the opposition, but I wish that Locomotive had moved totally out of the steam age! – **R. Barker, Redcar.**

Computing Ltd

WORLD CUP FOOTBALL

AMSTRAD AND C16 £7.95

ALSO AVAILABLE FOR CBM 64 AND SPECTRUM 48K.

WORLD CUP

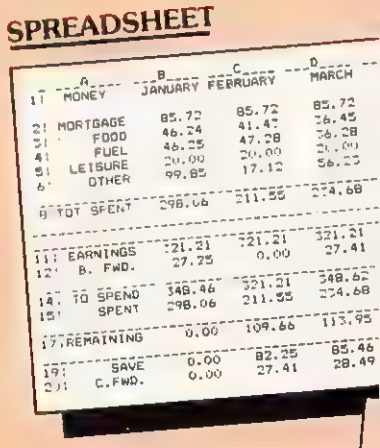
Can you lead your team to the World Cup? Up to nine players can play, each with a different team (country). Choose your team from a menu of countries, then prepare to play. Guide your men around the 3D pitch, and watch the crowd cheer as you score a goal! Depending on which team you are up against you will either play an opponent or the computer. Knock out all the other teams and you become the World Champions. Amazing animated action.

ARTIC COMPUTING LTD, MAIN STREET, BRANDESBURTON, DRIFFIELD YO25 8RL 0401 43553

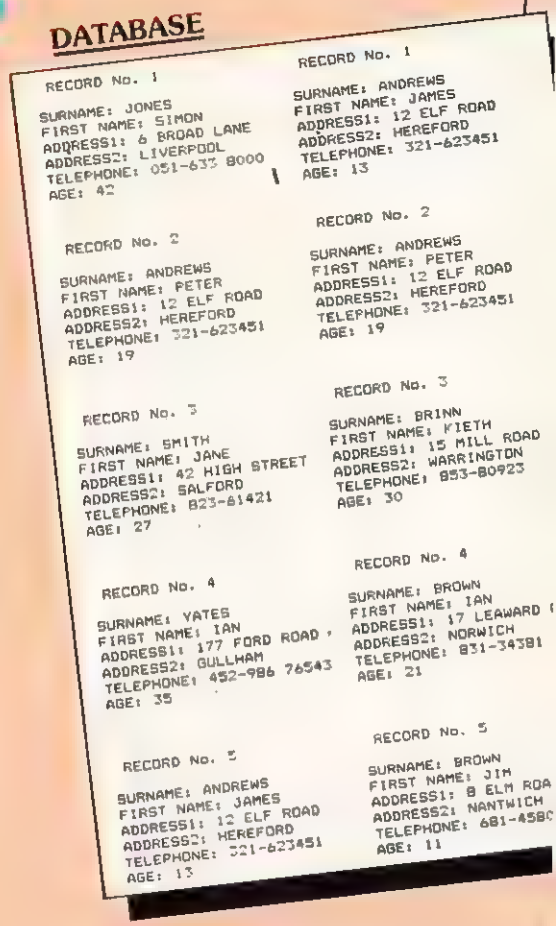
QUICK TO LEAR

THAT'S...

MINI OFFICE



JUST LOOK WHAT THIS PACKAGE CAN DO!



WORD PROCESSOR – Ideal for writing letters or reports! *Features:* Constant time display ● Constant word count (even shows words per minute) ● Normal or double-height text on screen or printout.

SPREADSHEET – Use your micro to manage your money! *Features:* Number display in rows and columns ● Continuous updating ● Update instantly reflected throughout spreadsheet ● Save results for future amendments.

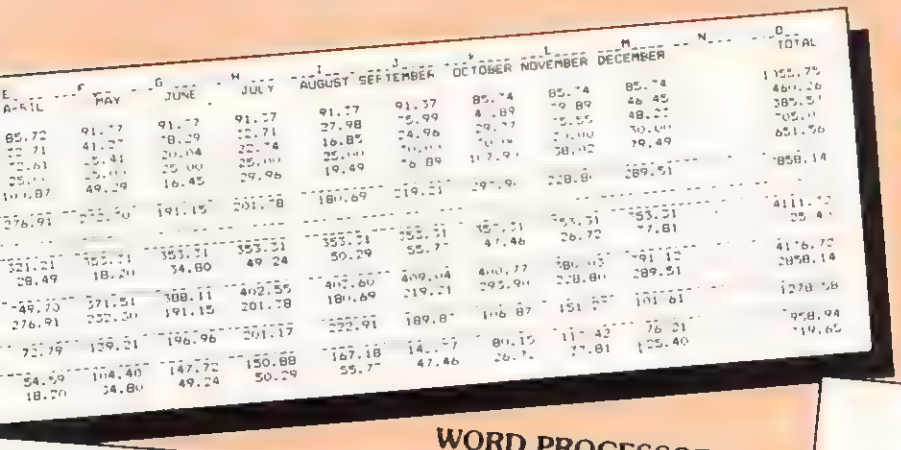
GRAPHICS – Turn those numbers into an exciting visual display! *Features:* 3D bar chart ● Pie chart ● Graph.

DATABASE – Use it like an office filing cabinet! *Features:* Retrieve files at a keystroke ● Sort ● Replace ● Save ● Print ● Search.

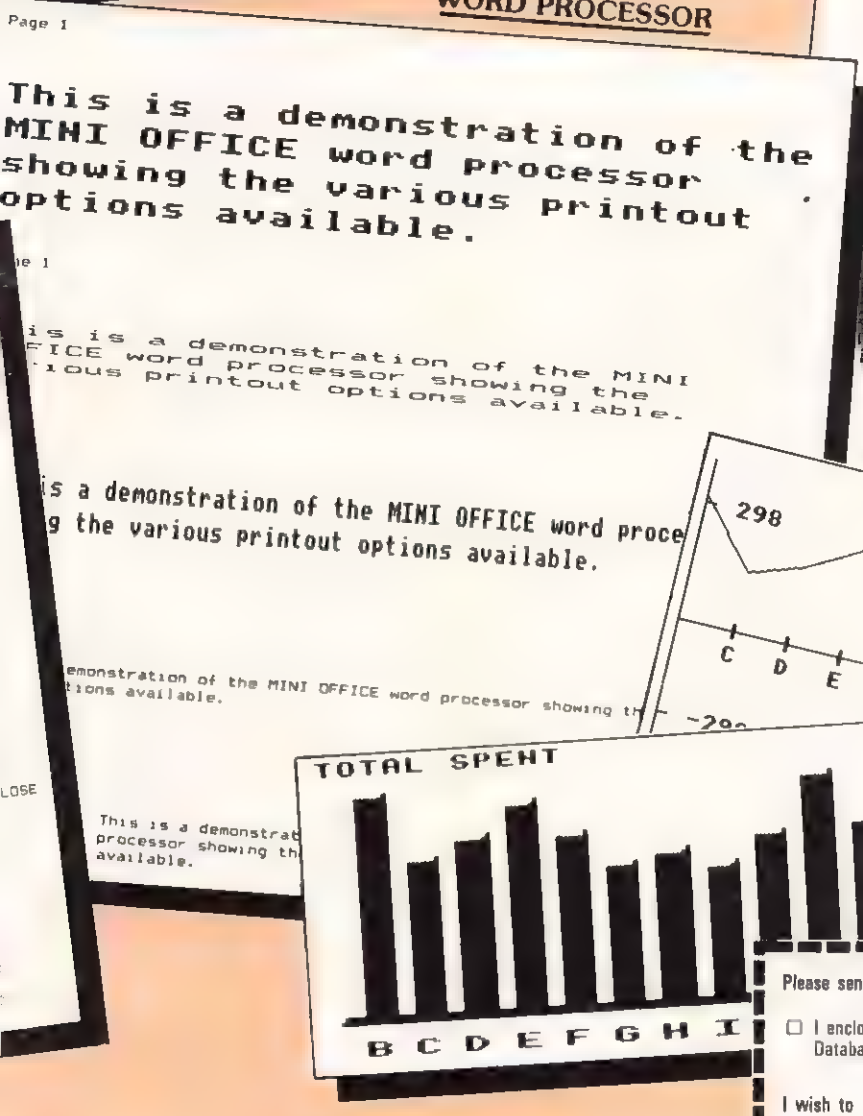
...and it's all at price of just

RN, EASY TO USE

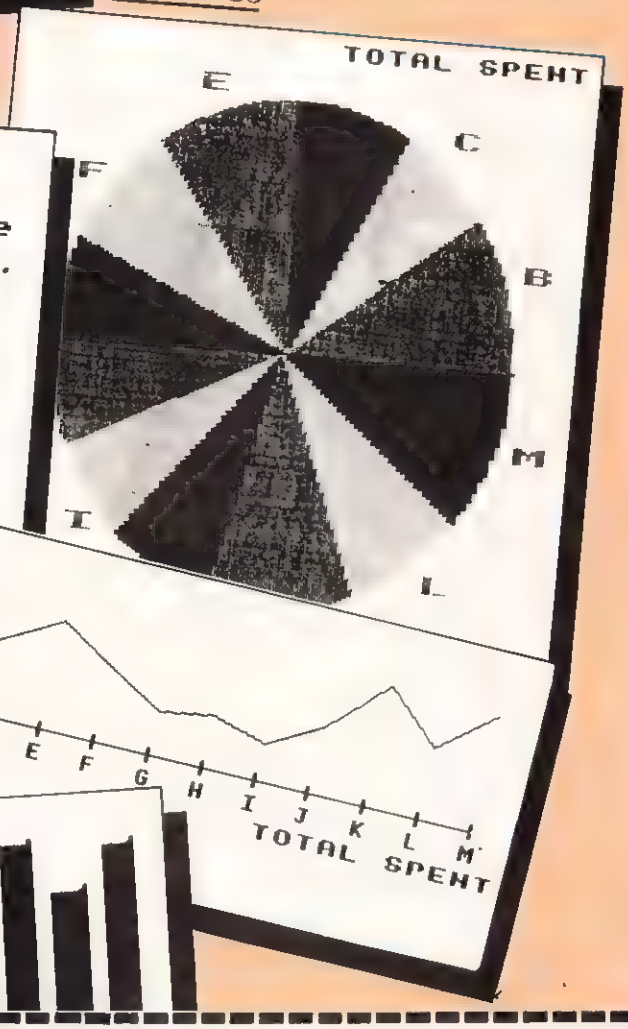
Specially written for your AMSTRAD CPC 464



WORD PROCESSOR



GRAPHICS



the unbelievable

£5.95
CASSETTE
3" DISC £9.95

DATABASE SOFTWARE

Please send me a copy of Mini Office for my Amstrad CPC464

☐ I enclose my cheque made payable to Database Publications Ltd.

☐ Amstrad cassette	£5.95
☐ Amstrad 3" disc	£9.95

Please tick box

I wish to pay by
☐ Access ☐ Visa No. _____ Expiry date _____

Signed _____

Name _____

Address _____

POST TO: Mini-Office Offer, Database Software,
68 Chester Road, Hazel Grove, Stockport SK7 5NY.

CA6

You're never too young to play a Magical Adventure on the Amstrad CPC464 . . .

Based on the style of the classic computer adventures – but written so that even small children can learn to find their way around, encouraged by colourful graphics and exciting sound effects.

The pack contains a 48-page full colour storybook
PLUS
a full length multi-location adventure on cassette for only

£8.95! post free

Read the book – then play the game!

Makes an ideal present

Please send me the complete Magic Sword pack for the Amstrad CPC464 containing storybook and cassette to:

Name _____

Address _____

_____

_____

☐ I enclose my cheque for £8.95 payable to Database Publications
☐ Or debit my Access/Visa card:

No. _____

Signed _____

SEND TO: **Adventure offer, Europa House, 68 Chester Road, Hazel Grove, Stockport SK7 5NY**

CA6

Now you've started computing with the Amstrad you'll want to read EVERY issue!

FREE! Send in your subscription on the form below and we'll send you this month's cassette, worth £3.75, entirely free of charge!

Give your fingers a rest

All the program listings from each month's issue are available for only **£3.75** *on cassette and* **£6.75** *on disc.*

JANUARY: ***Smiley:*** *Guide Smiley round the labyrinth avoiding Grumpies.* ***Code:*** *Be a mastermind.* ***Binary:*** *Baffled by binary bits? Let our utility help you out.* ***Dancer:*** *Simple but fun. This is a lovely mover.* ***Trapper:*** *Pen the monsters of the maze.* ***Scroller:*** *A slick sideways scrolling routine.* ***Letter Litter:*** *Learn the keyboard.*

FEBRUARY: ***Digger:*** *Search for crystals and avoid the aliens.* ***Simon:*** *Our version of this classic game.* ***Kingdom of Craal:*** *A devious fantasy.* ***Text Editor:*** *A powerful word processor.* ***Reaction Timer:*** *Test out your reactions on this simple little program.* ***Origins:*** *Changing the graphics origin produces dramatic effects.*

MARCH: ***Starfleet:*** *Can you qualify as a starfleet pilot?* ***Swamp:*** *Join the battle against giant frogs.* ***Parachute:*** *Parachute to earth in this educational program.* ***Dump:*** *Screen dump utility for Modes 0 and 1.* ***Hexer:*** *A hexadecimal loader.* ***PLUS . . . 3D Four in a Row.***

APRIL: ***Mad Adder:*** *Appease our aggravated adder.* ***Egg Blitz:*** *Eggs, boy scouts and a stunt pilot make up the fun.* ***Simple Sprites:*** *Create sprite-like effects.* ***Caesar Cipher:*** *Crack the code.* ***Pilot:*** *An easy to learn CAL language.* ***Spelling:*** *Learn to spell with this educational game.* ***Aleatoire:*** *The odd ways mariners make decisions.* ***Palindrome Tester:*** *Slick string handling techniques.*

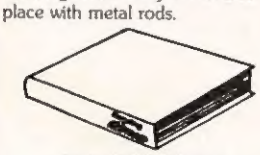
MAY: ***Tronn Cycles:*** *Enter the Maze and pit your wits in this two player arcade classic.* ***Castle of Fear:*** *Fantasy is the order of the day in this baffling concoction.* ***Mouse:*** *Can you trap the furry fellow in this simple but addictive game?* ***Cedric:*** *Fun with shape recognition in this novel interpretation of pelmanism.* ***Character Maker:*** *Define your own characters with our superb utility.* ***RSX:*** *New commands for your micro.* ***Cards:*** *Baffle your friends with an apparently simple computer card trick.* ***Trees:*** *A clever routine to create 'trees'.*

JUNE: ***Easydraw:*** *Hours of fun and entertainment. This versatile graphics utility allows you to create and manipulate your own pictures, saving them to tape or disc.* ***Da Bells:*** *Hilarious arcade action for the whole family with the obligatory Hunchback on the ramparts. Can you rescue Esmerelda?* ***AMmon:*** *A superb machine code monitor to complement our Z80 series.* ***Menu:*** *Create colourful menus with this intelligent disc utility.* ***Nim:*** *Baffle your friends with Alleatoire's simple routine to play the famous mathematical game.* ***Quads:*** *A clever routine to display expanding quadrilaterals.*

New! We are able to offer you annual subscription for the monthly listings cassette for only £40, and on 3" disc for only £70.

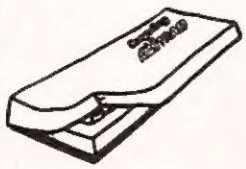
How to keep your collection complete

Bound in rich burgandy pvc and bearing the Computing with the Amstrad logo, this handsome binder will hold a year's supply of the magazines firmly secured in place with metal rods.



Only £3.95 (UK)

Protect your keyboard with our luxury dust cover made of soft, pliable, clear and water-resistant vinyl, bound with strong cotton and decorated with the magazine's logo.



Dust Cover Only £3.95

ORDER FORM

All prices include postage, packing and VAT and are valid to July 31

Please enter your requirements by ticking boxes — £ p

Annual subscription

UK £12	6001
EIRE £13 (IR £16)	6002
Overseas (Surface) £20	6003
Overseas (Airmail) £40	6004

Please send me my free May issue listings cassette ☐

Commence with _____ issue — TOTAL _____

Back issues

£1.25 UK
£1.50 Overseas (Surface)

January	6006
February	6007
March	6008
April	6009
May	6010

TOTAL _____

Tapes

£3.75 (UK & Overseas)

January (Smiley)	6020
February (Digger)	6021
March (Starfleet)	6022
April (Mad Adder)	6023
May (Tronn Cycles)	6024
June (Da Bells)	6025

TOTAL _____

3" Discs

£6.75 (UK & Overseas)

January (Smiley)	6060
February (Digger)	6061
March (Starfleet)	6062
April (Mad Adder)	6063
May (Tronn Cycles)	6064
June (Da Bells)	6065

TOTAL _____

Cassette tape and disc annual subscription

Tape £40 (UK & Overseas)	6005
3" Disc £70 (UK & Overseas)	6059

Commence with _____ tape/disc (state month) — TOTAL _____

Dust Cover

£3.95 (UK & Overseas) — 6037 ☐ — TOTAL _____

Binder

£3.95 UK
£5.00 Overseas — 6038 ☐ — TOTAL _____

Payment: please indicate method (✓) — **TOTAL** _____

☐ Access/Mastercharge/Eurocard

Card No. ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐

☐ Barclaycard/Visa

Card No. ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐

☐ Cheque/PO made payable to Database Publications Ltd.

Name _____ Signed _____

Address _____

Send to:
***Computing with the Amstrad*, FREEPOST, Europa House, 68 Chester Road, Hazel Grove, Stockport SK7 5NY.**
(No stamp needed if posted in UK) *Please allow 28 days for delivery*

YOU CAN ALSO ORDER BY PHONE:

061-480 0171
(24 hours)

Don't forget to quote your credit card number and full address

POOLSWINNER

THE ULTIMATE POOLS PREDICTION PROGRAM

Boxed, with detailed instruction booklet

- **MASSIVE DATABASE** Poolswinner is a sophisticated Pools prediction aid. It comes complete with the largest database available – 22000 matches over 10 years. The database updates automatically as results come in.
- **PREDICTS** Not just SCOREDRAWS, but AWAYS, HOMES and NO SCORES.
- **SUCCESSFUL** SELEC guarantee that Poolswinner performs significantly better than chance.
- **ADAPTABLE** Probabilities are given on every fixture – choose as many selections as you need for your bet. The precise prediction formula can be set by the user – you can develop and test your own unique method.
- **SIMPLE DATA ENTRY** All English and Scottish team names are in the program. Simply type in the reference numbers from the screen. Or use FIXGEN to produce fixture list automatically (see below).
- **DISC/MICRODRIVE COMPATIBLE** All versions (except Apple and IBM) are supplied on tape, with simple instructions for conversion to disc/microdrive operation.

(This seasons results are supplied with the package so that predictions can start immediately.)

AVAILABLE FOR Spectrum (48K), Commodore 64, VIC 20 (+16K), AMSTRAD CPC 464, BBC B, Atari (48K), ZX81 (16K), Dragon, Apple II, IBM pc, ELECTRON

PRICE £15.00 (all inclusive)

NEW

FIXGEN 84/5

AT LAST: No more struggling for hours to get the fixture list into the computer. FIXGEN has been programmed with all English and Scottish fixtures for 1984/5. Simply type in the date, and the full fixture list is generated in seconds. Fully compatible with Poolswinner.

POOLSWINNER with FIXGEN £16.50 (all inclusive)

Fixgen alone £5.50 (yearly updates available)

COURSEWINNER V3

THE PUNTERS COMPUTER PROGRAM

Coursewinner is designed to allow you to develop and test your own unique winning system. Using information from daily newspapers or 'Sporting Life', the most important factors can be input and analysed. The program is supplied with a database detailing best trainers and jockeys, and effect of the draw for all British courses. (Flat & National Hunt.)

AVAILABLE FOR Spectrum (48K), Commodore 64, BBC (B), AMSTRAD CPC 464, Atari (48K), Apple II

PRICE £15.00 (all inclusive)

AVAILABLE (RETURN OF POST) FROM . . .

VISA phone 24 hrs

selec SOFTWARE

Access phone 24 hrs

37 COUNCILLOR LANE, CHEADLE, CHESHIRE. ☎ 061-428 7425

ADVERTISERS INDEX

AN ADVANTAGE AT LAST!

JOIN THE INDEPENDENT USER GROUP FOR THE AMSTRAD CPC464

For the same price as one games cassette, you get:

- 12 monthly newsletters with ideas, reviews, letters and more.
- Access to a database of information, programs and advice.
- Opportunities to communicate with other computer users.
- Offers of reasonably priced software and accessories.

Send £8.95 to:
ADVANTAGE, 33 Malyns Close,
Chinnor, Oxfordshire

or send large stamped addressed envelope for more information

- MAXAM -
FOR THE AMSTRAD

THE COMPLETE CODE DEVELOPMENT SYSTEM FOR THE AMSTRAD CPC 464.

★ ASSEMBLER ★ MONITOR ★ TEXT EDITOR ★

"The Arnor system is the best editor/assembler to be released for the AMSTAD so far" - PCN 100
"For flexibility and ease of use, ARNOR is easily the best I have seen" Pop. C. Wkly Vol 4 No 8
"assemblers.... look no further, ARNOR's is the best I have seen... by far the easiest to use and most friendly I have come across"
- Computing with the Amstrad. April 1985.

Now available in **ALL THREE** formats

Tape (only) £13.50

Disc £26.90.

16K ROM + multifunction adaptor £59.90

(All prices include VAT, p & p)

Cheques/Po's to ARNOR Ltd Dept □
PO BOX 619, London SE25 6JL

-Overseas - no extra - Trade enq's welcome -

Make MAXimum use of your AMstrad

(CA)

Hotline 01.653.1483 **Technical 01.852.2174**

SIMPLE ACCOUNTS
AMSTRAD CPC 464

- FULL ANALYSIS OF INCOME & EXPENDITURE
- POWERFUL ENTRY SEARCH ROUTINES
- CONSTANT UPDATE OF CREDITORS AND DEBTORS
- SPECIAL IN-DEPTH ANALYSIS OPTION
- VAT REPORT ● DATE SORT
- MONTHLY AND YEAR TO DATE REPORTS
- FULLY DOCUMENTED ● EASY TO USE

ONLY £29.95 including VAT and carriage

Full specification manual and worked examples available on request.

CORNIX SOFTWARE LTD
16 Kneesworth Street, Royston, Herts. SG8 5AA.
Tel: Royston (0763) 46065

Specialists in business and financial software for the serious user.

SHEKHANA COMPUTER SERVICES

	RRP	Our Price		RRP	Our Price
Ghostbusters	10.99	9.99	Hunchback II	8.95	7.75
American Football	9.99	8.25	Daley Toms Decathlon	8.95	7.75
Sorcery	8.95	7.90	Kong Strikes Back	8.95	7.75
Master Chess	8.95	7.50	Steve Davis Snooker	7.95	6.50
Blagger	8.95	7.90	Classic Racing	8.95	7.75
Dark Star	7.95	6.75	Technician Ted	7.95	6.90
Gilligans Gold	6.95	6.25	Combat Lynx	8.95	8.00
Flight Path 737	6.95	5.90	Forest at Worlds End	6.00	4.75
Football Manager	7.95	6.75	Message from Andromeda	6.00	4.75
Sir Lancelot	6.95	5.90	Jewels of Babylon	6.00	4.75
Fighter Pilot	8.95	7.50	Heroes of Karn	6.00	4.75
Jet Set Willy	8.95	7.50	Chopper Squad	6.00	4.75
Manic Miner	8.95	7.50	Azimuth (Head Alignment)	8.99	7.95
The Hobbit	14.95	12.50	Flight Sim (Myrddin)	11.95	10.90
Defend or Die	7.95	6.50	Guide to Basic Part I	19.95	15.00
Pyjamarama	8.95	7.50	All Level 9 Games	9.95	8.25

Cheques/PO to:

SHEKHANA COMPUTER SERVICES
653 Green Lanes, London N8 0QY

(Mail Order address only). Tel: 01-800 3156. (SAE for List) – or visit our shop at Marbles Shopping Centre, Unit 11, 527-531 Oxford Street W1R 1DD. Open 7 days a week from 10am – 19.00pm (Opps. Marble Arch Tube Station)

Above discounts applicable only on production of this advert at our shop

Amstrad CPC464 Speech Synthesizer

The dk'tronics Amstrad speech synthesizer and powerful stereo amplifier uses the popular SLO/256 speech chip and has an almost infinite vocabulary. It is supplied with a text to speech converter for ease of speech output creation. Everything you wish to be spoken is entered in normal English, without special control codes or characters, it is therefore extremely easy to use. The voicing of the words is completely user transparent and the computer can carry on its normal running of a program while the speech chip is talking. The speech output from SLO/256 is mono and directed to both speakers.

Stereo Output

To utilise the Amstrad stereo output on the back of the computer, the interface has a built in stereo amplifier, this gives all sound output a totally new dimension and greatly improves the sound quality and volume over the computer's internal speaker. Any sound that previously came out of the mono speaker will now be sent out via the interface in stereo. All programs that use the sound in anyway (i.e. commercial software) will now output through the interface, which is fitted with volume and balance controls.

Speech Synthesis

The Amstrad speech synthesis utilises parts of the spoken word known as allophones. These are actual sounds that go to make up speech. The SP0256 allophone speech synthesis technique provides the ability to synthesize an almost unlimited vocabulary. Fifty-nine discrete speech sounds (allophones) and five pauses are stored in the speech chip's internal rom.

Text to Speech

Although there are only 26 letters in the alphabet, letters have a totally different sound when used in different words. For example, The "a" in 'Hay' is much longer and softer than in 'Hat'. When you speak you automatically make adjustments because you know just how a word should sound Not quite so easy with a computer.

The machine code software is mainly developed to this mode of operation. 3.5K is used for tables which contain the rules & exceptions to the rules of the English Language.

e.g. I before E except after C) This therefore allows the user to enter words to be spoken in normal English.

Speakers

Supplied with the Speech Synthesizer are two high quality 4" speakers these have been designed to compliment theAmstrad Computer. They are fitted with 1 metre of cable and can be positioned for the best stereo effect. The synthesizer interface fits neatly on to the rear of the computer. It has a through connector to enable other interfaces (e.g. Disc Drive) to connect to the rear of the synthesizer for ease of expansion Please send S.A.E. for a copy of the instruction manual which will give full and comprehensive details.

New Basic Commands

There are 8 new Basic Commands which control all the functions of the interface. Making the Synthesizer very easy to use. You can even control the speed at which it will talk to you. Or use the synthesizer to create sound effects like a fourth sound channel.

10 PRINT " 'AMSTRAD' "

The above is an example of the Syntax for entering speech into the computer and shows how simple it is to use.

The instruction book gives comprehensive details and examples of how to use the interface both from machine code and basic.

How to Order

The Amstrad Speech Synthesizer costs only £39.95. You can obtain your synthesizer through any good computer store or by completing the order form and returning it to:
dk'tronics Limited, Shire Hill, Saffron Walden, Essex. OR by telephone quoting your barclaycard or access number. Orders normally despatched within 24 hours.

Please rush me

.....[QTY] Amstrad Speech Synthesizer at £39.95 +£1.25 p&p
I enclose cheque/PO/Cash for Total £.....
or debit my Access/Barclaycard No.

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Signature.....
Name.....
Address.....

dk'tronics

Saffron Walden, Essex CB11 3AQ
Tel: (0799) 26350 10 lines

the only choice

Kuma AMSTRAD CPC464 software

Holdfast | Gems of Stradus | Star Avengers | Galaxia | Music Composer

Logo | Database | ZEN Assembler | EASIVAT | Home Budget

An outstanding selection from Kuma's rapidly expanding range of Entertainment and Application Software for the Amstrad CPC 464 Micro-computer.

BOOK ● ***The Amstrad CPC 464 Explored.***

This superb book is designed to let every CPC 464 user, at whatever level, get the most from his computer. After an introductory section on the special Basic features, the book looks in depth at the excellent sound and graphic facilities.

Now available from selected branches of Co-op, Granada, LASKYS and John Menzies

Kuma Computers Ltd., Unit 12, Horseshoe Park, Horseshoe Road, Pangbourne, Berks RG8 7JW.

Please send full catalogue on Amstrad CPC464 products.

Name .....

Address .....

..... Phone.....

I own an Amstrad CPC 464 computer ☐

Trade Enquiries Phone 07357-4335

Visitors wishing to call at our Pangbourne Manufacturing and Distribution Centre are advised to phone 07357-4335 first for an early appointment.